Vol.12 No.5 October 1993

BEEBUG

FOR THE BBC MICRO & MASTER SERIES

Compact Disc Cataloguer

- WORDWISE-MAIL
- A ROOM WITH A VIEW
- QUASIMODO
- MULTIPLE CHARACTER DESIGNER

BEEBUG Vol.12 No.5 October 1993

FEATURES

REVIEWS

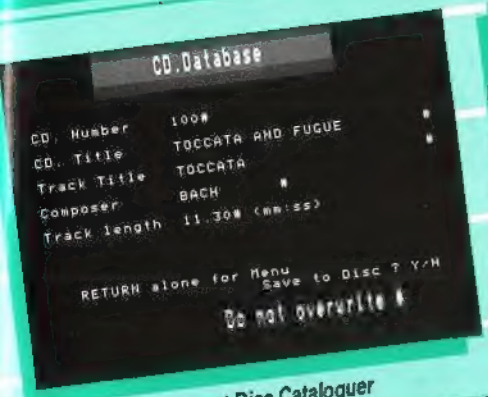
REGULAR ITEMS

HINTS & TIPS

PROGRAM INFORMATION

All listings published in BEEBUG magazine are produced directly from working programs. They are formatted using LISTO 1 and WIDTH 40. The space following the line number is to aid readability only, and may be omitted when the program is typed in. However, the rest of each line should be entered exactly as printed, and checked carefully. When entering a listing, pay special attention to the difference between the digit one and a lower case l (L). Also note that the vertical bar character (Shift \) is reproduced in listings as |.

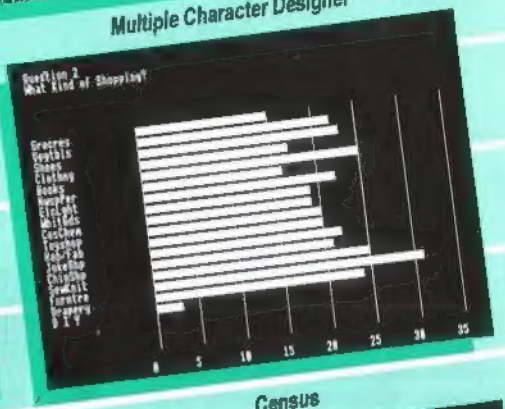
All programs in BEEBUG magazine will run on any BBC micro with Basic II or later, unless otherwise indicated. Members with Basic I are referred to the article on page 44 of BEEBUG Vol.7 No.2 (reprints



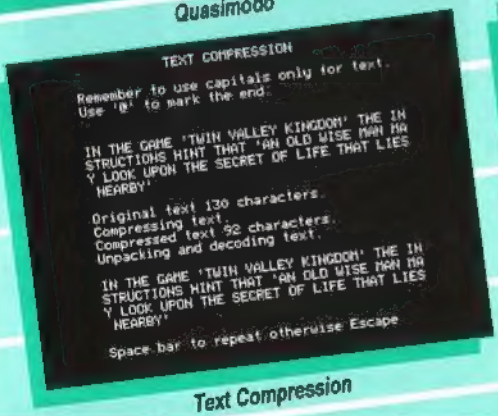
Compact Disc Cataloguer

Multiple Character Designer

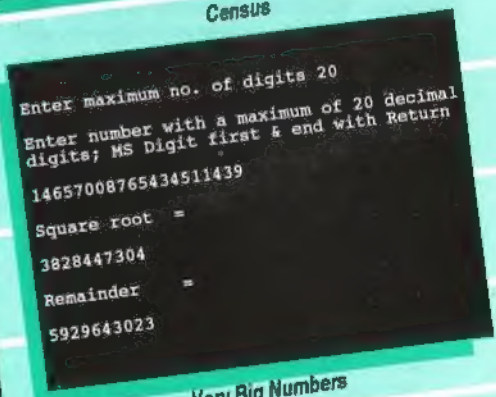
Quasimodo



Census



Text Compression



Very Big Numbers

available on receipt of an A5 SAE), and are strongly advised to upgrade to Basic II. Any second processor fitted to the computer should be turned off before the programs are run.

Where a program requires a certain configuration, this is indicated by symbols at the beginning of the article (as shown opposite). Any other requirements are referred to explicitly in the text of the article.

Program needs at least one bank of sideways RAM.

Program is for Master 128 and Compact only.

Editor's Jottings/News

ACORN WORLD '93

By the time you read this issue of BEEBUG, Acorn World '93 will be little more than four weeks or so away. After years of rumour regarding Acorn's involvement in shows organised by other sponsors (principally Acorn User), Acorn took the bull by the horns last October and announced that the 1993 autumn show would be organised directly by themselves. Thus Acorn World '93 was born.

Advance information from Acorn does seem to indicate that this autumn we can expect something different. Acorn is clearly keen to promote its systems in specific targeted markets, notably education (which we all know anyway), in the consumer (i.e. games) market, and increasingly in the professional DTP market where Acorn's Archimedes system is competing against the more established Apple Macintosh.

In the consumer market, Acorn will be launching a major sales campaign this autumn backed up by some aggressive pricing (their words)! What this means in practice is that there are significant price reductions to be had on most Archimedes models. The A3010 Family Solution, previously £499 inc. VAT will be reduced to £399 inc. VAT, and there are similar price reductions on other models. The result is that the cost of buying a new Archimedes system is now lower than it has ever been, and no doubt the asking price on secondhand machines will follow suit.

Acorn has also been making a number of announcements this year about links with major publishing system suppliers. At Acorn World you can expect to see a new generation of DTP software pushing the Archimedes firmly into contention at the top end of this market. Acorn promise a complete publishing area at Acorn World, demonstrating all the processes involved in editing and printing a major publication - they even promise a full colour press with copy coming off the production line.

Acorn World will also feature a theatre, with a programme of lectures on each of the three days, but unlike previous shows, where speakers have extolled the virtues of one commercial software solution after another with a strong sales bias, Acorn has promised a line-up of quality speakers, addressing the issues that face the Acorn community, in what truly promises to be a 'Vision for the Future', Acorn's catch-phrase for the show.

Now no doubt many of you reading this may be feeling that this is just another editorial pushing the Archimedes, but that is not my immediate aim. Whether you have a simple model B, a Master 128, or a host of add-ons like co-processors, we are all users of Acorn machines. And Acorn World promises to be the most interesting and exciting event for all Acorn users for many years. If you have a chance then come along and see what Acorn, and all the other exhibitors, are up to.

We would also like to welcome you to any of our three stands: RISC User (and BEEBUG) where you will be able to meet editorial staff; RISC Developments Ltd. for our ranges of software and hardware; and Beebug Ltd. which is still one of the largest and most respected Acorn dealers in the country. See you at Acorn World '93.

Mike Williams

Compact Disc Cataloguer

Keep track of all your compact discs with this handy application from Graham Lowe.

Following the recent purchase of a compact disc player, I started building a collection of classical music CD's. It quickly became obvious that some sort of filing system would be useful to keep them in some semblance of order, particularly compilation discs, which have many items on them each by a different composer. It occurred to me that a program I wrote several years ago to record cheque transactions could be modified to suit this purpose. The program presented here is the result.

Searching for a composer

As it stands it is pretty skeletal, with scope for expansion or alteration for other uses. The programming is, I suspect, a bit archaic, and not necessarily the tidiest code ever written, although it is quite easy to follow. Perhaps other readers may have suggestions for improving the code.

The data itself is stored on disc in an extending file called *cd_data*. Since each new entry is added at the end of the file, there is no scope for editing an entry after it has been saved. So a prompt is given before saving to give you the opportunity for checking your data.

SETTING UP THE CD CATALOGUER

Type in the *cd_base* program and save it to disc. Then create an empty data file with:

```
X%=OPENOUT("cd_data")
CLOSE#X%
```

This file must be the last file on the disc, so that it can expand without the dreaded "Can't Extend" error appearing. If for some reason this error should occur you should:

*COPY *cd_*data to another disc
*COMPACT the original disc and
*COPY *cd_*data back again

Should more than one data file be set up, the current one (the one which you will be adding data to) must still be the last on the disc.

A very productive fellow this Bach

USING THE CATALOGEUR

When run the program displays a menu, which presents you with the usual facilities for adding details, and for searching using the CD number, CD title, track title, composer or track length as the search criteria. The CD number is a number assigned by you, when you enter the original details. When entering data don't overwrite the hash symbol. Doing so will not have far reaching

consequences but will spoil the display during searches. Also avoid using punctuation as strings are cut short by commas and quotes. Selecting "K" from the main menu calls *PROCkeys* which allows you to insert a string into f0. I have found during input that with a multi-track CD it becomes very irritating typing in the CD title 15-20 times, so having the main title typed into f0 can be useful. *PROCkeys* could easily be expanded to use more than one function key. "Q" as always, quits the program.

The main menu

HOW THE PROGRAM WORKS

PROCinput is pretty straightforward. Simply a series of input commands and messages. *PROCscreen(message$)* is responsible for printing the basic screen for each search section. To modify the procedure for other uses simply change *message$*. *PROCsearch(field%,find$)* uses the INSTR function to test the data strings in groups of five. The variable *field%* defines which of the five is to be tested for a match with *find$*. I use all upper case for input to avoid missing lower case entries during searches. Using INSTR means a search can be made with only part strings though the results can be a bit odd. A search for Composer using just "BA" will find not only Bach and Bartok but also Offenbach. The longer *find$* is, the narrower the field of results will be. Try to keep the input data as constant as possible, especially abbreviations. This program doesn't break any speed records with

Entering a new CD

sophisticated search techniques; it is simply a linear search from one end of the file to the other. When the search time becomes too long, additional data files could be started, though a facility to select a file name would then be required. *PROCquery* simply waits for confirmation before saving to disc. *PROCding* changes the VDU7 sound courtesy of the pages of BEEBUG.

I hope this program may prove useful in helping some readers tame their otherwise untidy CD collections.

```
  10 REM Program cd_base
  20 REM Version B 1.01
  30 REM Author  Graham Lowe
  40 REM BEEBUG  October 1993
  50 REM Program subject to copyright
  60 :
 100 MODE 7
 110 PROCding:CLS
 120 ON ERROR CLOSE#0:REPORT:PRINT" at
line ";ERL:END
 130 DIM Q$(5)
 140 exit%=FALSE
 150 REPEAT
 160 find$="":field%=0
```

```
  170 field%=FNchoose
  180 IF field%=27 PROCkeys ELSE IF fiel
d%=33 exit%=TRUE ELSE ON field% PROCinpu
t,PROCfindnum,PROCfindtitle,PROCfindt_ti
tle,PROCfindcomp,PROCfindtime
  190 IF find$<>"" THEN MODE0:VDU19,1,2;
0;:PROCsearch(field%,find$)
  200 MODE 7:CLOSE#0
  210 UNTIL exit%
  220 CLS:CLOSE#0
  230 END
  240 :
 1000 DEFFNchoose REM Sub menu screen
 1010 PROCheader1
 1020 VDU31,6,6,134,157,129:PRINT"1...Ad
d Details":VDU31,31,6,156
 1030 VDU31,0,9,134,157,129:PRINT"Find D
etails:-":VDU31,20,9,156
 1040 VDU31,6,12,134,157,129:PRINT"2...C
D.Number.":VDU31,31,12,156
 1050 VDU31,6,14,134,157,129:PRINT"3...T
itle.":VDU31,31,14,156
 1060 VDU31,6,16,134,157,129:PRINT"4...T
rack Title.":VDU31,31,16,156
 1070 VDU31,6,18,134,157,129:PRINT"5...C
omposer.":VDU31,31,18,156
 1080 VDU31,6,20,134,157,129:PRINT"6...T
rack Length":VDU31,31,20,156
 1090 VDU31,6,23,134,157,129:PRINT"Q...Q
UIT    K..KEYS":VDU31,31,23,156
 1100 REPEAT
 1110 G=GET
 1120 UNTIL((G>48 AND G<55) OR G=75) OR
G=81)
 1130 =G-48
 1140 :
 1150 DEFPROCdisc
 1160 X=OPENUP"ADFS::HardDisc.$.Steve.La
test.CD_Cat.cd_data"
 1170 PTR#X=EXT#X
 1180 PRINT#X,cd$,title$,t_title$,comp$,
len$
 1190 CLOSE#X
 1200 ENDPROC
 1210 :
 1220 DEFPROCheader
 1230 PRINT"CD no"TAB(14)"TITLE"TAB(34)"
TRACK TITLE"TAB(59)"COMPOSER"TAB(73)"LEN
GTH":PRINTTAB(0,1)STRING$(80,"_"):PRINT
 1240 ENDPROC
 1250 :
 1260 DEFPROCprint
 1270 PRINTTAB(0)Q$(1)TAB(6)Q$(2)TAB(33)
Q$(3)TAB(60)Q$(4)TAB(74)Q$(5)
 1280 ENDPROC
 1290 :
 1300 DEFPROCsearch(field%,find$)
 1310 VDU14:field%=field%-1
 1320 PROCheader
 1330 X=OPENUP"ADFS::HardDisc.$.Steve.La
test.CD_Cat.cd_data"
 1340 PTR#X=0:n%=0
 1350 REPEAT
 1360 INPUT#X,Q$(1),Q$(2),Q$(3),Q$(4),Q$
(5)
 1370 IF field%=5 n%=INSTR(LEFT$(Q$(fiel
d%),2),find$) ELSE n%=INSTR(Q$(field%),f
ind$)
 1380 IF n%>0 PROCprint
 1390 UNTIL EOF#X
 1400 CLOSE#X
 1410 PRINT''TAB(40);"P R E S S  S P A C
E"
 1420 PRINTTAB(40);STRING$(20,"_")
 1430 REPEATUNTILGET=32
 1440 VDU15
 1450 ENDPROC
 1460 :
 1470 DEFPROCfindnum
 1480 CLS
 1490 PROCscreen("CD_Number Search")
 1500 VDU31,0,12,131,157,132,:PRINT"CD N
umber Required..." TAB(27,13)">RETURN<":
INPUTTAB(24,12)find$
 1510 ENDPROC
 1520 :
 1530 DEFPROCfindtitle
 1540 CLS
 1550 PROCscreen("CD Title Search")
 1560 VDU31,0,12,131,157,132,:PRINT"Titl
e Required..." TAB(27,13)">RETURN<":INPU
TTAB(20,12)find$
 1570 ENDPROC
 1580 :
 1590 DEFPROCfindt_title
 1600 CLS
 1610 PROCscreen("Track Search")
 1620 VDU31,0,12,131,157,132,:PRINT"Trac
k Title..." TAB(27,13)">RETURN<":INPUTTA
B(18,12)find$
```

```
 1630 ENDPROC
 1640 :
 1650 DEFPROCfindcomp
 1660 CLS
 1670 PROCscreen("Composer Search")
 1680 VDU31,0,12,131,157,132,:PRINT"Comp
osers Name..." TAB(27,13)">RETURN<":INPU
TTAB(20,12)find$
 1690 ENDPROC
 1700 :
 1710 DEFPROCfindtime
 1720 CLS
 1730 PROCscreen("Track time search")
 1740 VDU31,0,12,131,157,132,:PRINT"Time
 required (in minutes)..." TAB(27,13)">R
ETURN<":INPUTTAB(32,12)find$
 1750 ENDPROC
 1760 :
 1770 DEFPROCheader1
 1780 CLS
 1790 VDU31,6,0,135,157,135,31,31,0,156
 1800 VDU31,6,1,133,157,132,141:PRINT"
 CD.Database":VDU31,31,1,156
 1810 VDU31,6,2,133,157,132,141:PRINT"
 CD.Database":VDU31,31,2,156
 1820 VDU31,6,3,135,157,135,31,31,3,156
 1830 ENDPROC
 1840 :
 1850 DEFPROCinput
 1860 PROCheader1
 1870 PRINTTAB(0,6)"CD. Number "TAB(17)"
#"
 1880 PRINTTAB(0,8)"CD. Title "TAB(39)"#
"
 1890 PRINTTAB(0,10)"Track Title"TAB(39)
"#"
 1900 PRINTTAB(0,12)"Composer"TAB(24)"#"
 1910 PRINTTAB(0,14)"Track length"TAB(19
)"# (mm:ss)"
 1920 PRINTTAB(3,19)"RETURN alone for Me
nu"
 1930 PRINTTAB(16,22)CHR$141;"Do not ove
rwrite #"
 1940 PRINTTAB(16,23)CHR$141;"Do not ove
rwrite #"
 1950 INPUTTAB(14,6)cd$:IF cd$="" ENDPRO
C
 1960 INPUTTAB(14,8)title$:IF title$=""
ENDPROC
 1970 INPUTTAB(14,10)t_title$:IF t_title
$="" ENDPROC
 1980 INPUTTAB(14,12)comp$:IF comp$="" E
NDPROC
 1990 INPUTTAB(14,14)len$:IF len$="" END
PROC
 2000 PROCquery
 2010 ENDPROC
 2020 :
 2030 DEFPROCquery
 2040 VDU7:PRINTTAB(20,20)CHR$136"Save t
o Disc ? Y/N"
 2050 reply$=GET$:IF reply$="Y" OR reply
$="y" PROCdisc:ENDPROC
 2060 ENDPROC
 2070 :
 2080 DEFPROCding
 2090 ENVELOPE1,1,0,0,0,0,0,0,-126,-2,0,
-1,126,80
 2100 *FX 211,1
 2110 *FX 212,0
 2120 *FX 213,180
 2130 *FX 214,1
 2140 ENDPROC
 2150 :
 2160 DEFPROCscreen(message$)
 2170 VDU31,6,0,133,157,132,141:PRINTmes
sage$:VDU31,34,0,156
 2180 VDU31,6,1,133,157,132,141:PRINTmes
sage$:VDU31,34,1,156
 2190 VDU31,0,20,132,157,134,:PRINT"PRES
S RETURN ONLY FOR MAIN MENU"
 2200 ENDPROC
 2210 :
 2220 DEFPROCkeys
 2230 CLS
 2240 PROCscreen("f_key Setup")
 2250 PRINTTAB(0,8)"Please type in the m
essage to be entered"'"in key f0. It mus
t not exceed the max"'"for the intended
 field."
 2260 PRINTTAB(0,14)"Cd_Number.....3"'"C
d_Title.....25"'"Track Title...25"'"Com
poser.....10"'"Time.....5"
 2270 INPUTTAB(0,19)fkey$
 2280 OSCLI("key0"+" "+fkey$)
 2290 ENDPROC
 2300 :
```

Multiple Character Designer

Roger Butler presents the answer to the VDU23 blues.

How often have you tried to design a large graphic character using multiple VDU23 commands, only to discover that the final result bears no relationship to your original concept? The following program provides a solution to these problems. I had originally planned to write a single character editor, but it did not take me long to realise that it was not worth the effort - you may as well use a piece of graph paper and save all that typing! So, I decided to write a multiple character editor instead, where you can see exactly what you are designing.

USING THE PROGRAM

Firstly, type in the listing as shown. After checking and saving, run the program and you will be presented with the following options.

1: Edit characters
2: Dump characters

THE EDITOR

Selecting option 1 will put you in *Edit* mode. Here you will be required to enter the size of the grid and which character, if any, you want to appear in each 'cell'. The grid has a maximum limit of 9 characters in a 3 * 3 format. However, other permutations are possible; for example: 3 rows by 1 column, 2 rows by 3 columns or 3 rows by 2 columns, etc. Let us suppose that you choose a grid size of (2,3); i.e. 2 rows by 3 columns = 6 characters to edit. You will need to enter a VDU character number for cells from (0,0) through to (1,2). If you are not familiar with matrix notation, the following diagram may help in showing which cell is which.

(0,0)	(0,1)	(0,2)
(1,0)	(1,1)	(1,2)

*Diagram 1. A simple 3*3 matrix*

So let's say for example you have chosen the following character numbers for each cell.

(0,0) - 65
(0,1) - 77
(0,2) - 37
(1,0) - 54
(1,1) - 84
(1,2) - 40

You would then be editing the following.

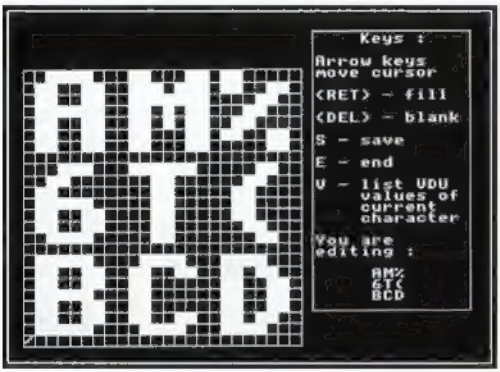
A	M	%
6	T	(

Diagram 2. 6 characters in 6 cells

If you are designing a graphic from scratch, you can simply enter the 'space' character 32 for each cell. After entering the character number for the last cell, the *editing* screen appears. On-screen instructions are provided, and are self explanatory.

After designing your masterpiece, you can have each character definition

displayed individually by pressing 'V' when the pointer is over the character. You will then have 20 seconds to write down the VDU23 command. You can cancel the display during this period by pressing any key. Pressing 'E' ends the editing session and gives you the option to list the VDU23 definitions on screen or printer.



The Editing screen

You can save your new characters by pressing 'S', which saves page &C00 to disc. To edit any saved characters, run the program, and then press Escape while in the editing screen. Type the command:

```
*LOAD filename
```

where *filename* is the name you saved the characters under. Now re-run the program and enter the number of each character you saved.

Readers will note the use of the *OSCLI* command in the program, which is not available on Basic 1. If you only have Basic 1, define a procedure as follows:

```
DEF PROCos($&740)
X%=&40 : Y%=7
CALL&FFF7:
ENDPROC
```

You can now use *PROCos()* in exactly the same way as OSCLI.

THE DUMP ROUTINE

You can use this option to begin with, or after pressing 'E' when in the editing mode. You will be prompted to enter the start and finish character numbers, after which the program will churn out the definitions of each VDU character within the range specified.

HOW THE PROGRAM WORKS

Like the options, the program consists of two main procedures, *PROCdump* and *PROCmove*. *PROCdump* is obviously the routine which prints the character definitions, and it does this by using an OSWORD call with A=10 (see the User Guide page 462). *PROCmove* is slightly more complicated. It uses the GET command to see which key has been pressed, and then acts upon this information. If a cursor key is pressed, the pointer is moved in the appropriate direction using *PROCdisp*. If 'S' is pressed, control is passed to *PROCsave* and then passed back again. *PROCfill* is the other routine within the *PROCmove* routine, and this is called when Return, Delete or 'V' is pressed. When Return is pressed, the computer checks to see if the square is already filled in. If it is, Return is ignored. If it is not, the square is filled and the character definition to which that square belongs is updated, as is the composite character in the instruction box. Delete works in exactly the same way as Return. When 'V' is pressed, *PROCfill* simply prints out the make-up of the character.

This program really has speeded up the design process for me and I am sure it will be of help to many readers.

```
   10 REM  Program Character Designer
   20 REM  Version B 1.0
   30 REM  Author  Roger Butler
   40 REM  BEEBUG  October 1993
   50 REM  Program subject to copyright
   60 :
  100 MODE1
  110 ON ERROR PROCerror:END
  120 VDU23;8202;0;0;0;19,1,5;0;0;19,3,6
;0;0;
  130 PROCinit
  140 PROCenter
  150 VDU19,1,1;0;0;
  160 PROCqa
  170 CLS
  180 GCOL0,2
  190 PROCgrid(r,c)
  200 PROCinst
  210 x1%=0:y1%=S%*x%=0:y%-S%
  220 PROCfsq
  230 VDU5
  240 PROCmove
  250 CLS
  260 PRINT'"Do you want to dump charact
ers (Y/N) ?"
  270 U=GET
  280 IF U=89 OR U=121 PROCdump
  290 PRINT '"Do you want to continue ed
iting (Y/N) ?"
  300 G=GET
  310 IF G=89 OR G=121 RUN
  320 END
  330 :
 1000 DEF PROCgrid(r,c)
 1010 FOR X%=0 TO H% STEP S%
 1020 MOVE0,X%
 1030 DRAWW%,X%
 1040 NEXT
 1050 FOR X%=0 TO W% STEP S%
 1060 MOVEX%,0
 1070 DRAWX%,H%
 1080 NEXT
 1090 MOVE825,100:DRAW1279,100:DRAW1279,
900
 1100 DRAW825,900:DRAW825,100
 1110 ENDPROC
 1120 :
 1130 DEF PROCinit
 1140 COLOUR2
 1150 PRINT''SPC(5)"Multiple character e
ditor"
 1160 PRINT"SPC(18)"by Roger Butler."
 1170 COLOUR1
 1180 PRINT'''SPC(10)"1: Edit characters
":PRINT''SPC(10)"2: Dump characters"
 1190 COLOUR3
 1200 PRINTTAB(5,20)"Press the number of
 your choice."
 1210 DIM h(2,2),w(2,2),s(2,2)
 1220 IF GET=50PROCdump
 1230 FORx=0TO2
 1240 FORy=0TO2
 1250 READZ,Y,S
 1260 w(x,y)=Z:h(x,y)=Y:s(x,y)=S
 1270 NEXT,
 1280 ENDPROC
 1290 :
 1300 DEF PROCenter
 1310 REPEAT
 1320 CLS
 1330 PRINT''''Character Editor":COLOUR2
 1340 PRINT''"Please enter grid size (Max
. 3*3)"
 1350 INPUT''"Number of rows ";r
 1360 INPUT''"Number of columns ";c
 1370 PRINT''"Grid size chosen (";r;",";c
;")"
 1380 INPUT''"Is this o.k. (Y/N) ";k$
 1390 UNTIL INSTR("Yy",k$)
 1400 DIM q%(8*r,c-1)
 1410 r=r-1:c=c-1
 1420 DIMF%(r,c)
 1430 FORqw=0TOr:FORqw1=0TOc:PRINT"Char
acter to be edited at (";qw;",";qw1;") "
;:INPUTo%:F%(qw,qw1)=o% NEXT,
 1440 S%=s(r,c).H%=h(r,c):W%=w(r,c)
 1450 ENDPROC
 1460 :
 1470 DEF PROCdisp
 1480 GCOL2,1
```

```
1490 MOVEx%,y%:PRINT"/"
1500 GCOL1,2
1510 MOVEx1%,y1%:PRINT"/"
1520 x%=x1%:y%=y1%
1530 ENDPROC
1540 :
1550 DEF PROCmove
1560 *FX4,1
1570 REPEAT
1580 *FX15,1
1590 PROCdisp
1600 G=GET
1610 IFG=86 PROCfill(x1%,y1%,1,1)
1620 IFG=13 PROCfill(x1%,y1%,1,0)
1630 IFG=127 PROCfill(x1%,y1%,0,0)
1640 IFG=137 x1%=x%-S%*(x%<W%-3*S%/2)
1650 IFG=136 x1%=x%+S%*(x%>=S%)
1660 IFG=138 y1%=y%+S%*(y%>S%)
1670 IFG=139 y1%=y%-S%*(y%<H%-S%/2)
1680 IFG=83 OR G=115 PROCsave
1690 UNTILG=69 OR G=101
1700 ENDPROC
1710 :
1720 DEF PROCfill(A%,B%,j,k)
1730 VDU4
1740 VDU28,0,5,10,2
1750 D%=A% DIVS%:E%=8*(r+1)-(B% DIVS%)
1760 GCOL0,j
1770 IF k=0:MOVEA%+4,B%-4:MOVEA%+S%-4,B
%-4:PLOT85,A%+4,B%-S%+4:PLOT85,A%+S%-4,B
%-S%+4
1780 VDU28,0,31,39,0
1790 IF k=0:IF j=1:IF (q%(E%,D% DIV8) A
ND 2^(7-(D% MOD8)))=0 q%(E%,D% DIV8)=q%(
E%,D% DIV8)+2^(7-(D% MOD8))
1800 IF k=0:IF j=0:IF (q%(E%,D% DIV8) A
ND 2^(7-(D% MOD8)))>0 q%(E%,D% DIV8)=q%(
E%,D% DIV8)-2^(7-(D% MOD8))
1810 E%=E% DIV8:D%=D% DIV8
1820 IFk=1 PRINTTAB(1,3)"VDU 23,";F%(E%
,D%);",";q%(8*E%,D%);",";q%(8*E%+1,D%);"
,";q%(8*E%+2,D%);",";TAB(5,4),q%(8*E%+3,
D%);",";q%(8*E%+4,D%);",";q%(8*E%+5,D%);
",";q%(8*E%+6,D%);",";q%(8*E%+7,D%);
1830 IFk=1:Y=INKEY(2000):PRINTTAB(1,3);
SPC(24);TAB(1,4);SPC(24);
1840 VDU23,F%(E%,D%),q%(8*E%,D%),q%(8*E
%+1,D%),q%(8*E%+2,D%),q%(8*E%+3,D%),q%(8
*E%+4,D%),q%(8*E%+5,D%),q%(8*E%+6,D%),q%
(8*E%+7,D%)
1850 COLOUR2:FORn=0TO·FORn1-0TOc
1860 PRINTTAB(31+n1,25+n);CHR$(F%(n,n1,
);:NEXT,
1870 VDU5
1880 ENDPROC
1890 :
1900 DEF PROCdump
1910 CLS
1920 INPUT''"Character to start from ";
s
1930 INPUT''"Character to finish at ";e
1940 PRINT''"Do you want the output sent
"·INPUT"to a printer too (Y/N) ";k$
1950 PRINT''"Use SHIFT to scroll."CHR$14
1960 IF k$="Y" OR k$="y" VDU2
1970 FORK=s TO e
1980 ?&70=K
1990 X%=&70:Y%=0 A%=&A·CALL &FFF1
2000 COLOUR2
2010 PRINT"V. 23,",K;
2020 FOR z=&71TO &78
2030 PRINT;",",?z,
2040 NEXTz:NEXTK
2050 VDU3:COLOUR3
2060 PRINT''"Press any key":b=GET
2070 CLS
2080 ENDPROC
2090 :
2100 DEF PROCgetchar(Z%,b2,b3)
2110 ?&70=Z%
2120 X%=&70:Y%=0:A%=10:CALL&FFF1
2130 FOR N%=0 TO 7
2140 q%(8*b2+N%,b3)=?(&71+N%)
2150 NEXT
2160 ENDPROC
2170 :
2180 DEF PROCqa
2190 FORb=0TOr
2200 FORb1=0TOc
```

Continued on page 15

ADFS and the 'E' Attribute

Ian Crawford unlocks that which should stay locked

In the good (bad?) old days when there were only the BBC Models A and B, and most normal people used cassettes to load and save programs, the advent of the disc drive was like a breath of fresh air. One soon learned to master the techniques and the only problems were the occasional accidental deletion of a program (or two) because one had forgotten (or was too lazy) to 'L'ock the file.

The advent of the Master series and the Archimedes running ADFS, presented different problems. As shown in the *Trouble Shooting Guide Part 4* by Gareth Leyshon, (BEEBUG Vol.11 No.10), the setting of various file and directory *attributes* can take many forms: (R)ead only, (W)rite only, (L)ock, (D)irectory (which is usually accompanied with the LR set), and finally the potentially disastrous (E)xecute attribute.

Now most people will *never* need or want to set data files or programs to 'E' because, once set, *it is impossible to remove*, according to the Acorn ADFS User Guide

SO WHAT?

Well consider this: You have a 'special' disc that has taken you years to develop with all your master programs, or data files that constantly have to be accessed and updated as part of a hobby, or your entire business list of clients names etc. Like me, you will *always* lock all those important files, but when you need to update information you obviously need to unlock them.

Due to laziness, I always use the **ACC. * WR* method which unlocks *all* files. Then I re-save the modified data and re-lock using **ACC. * LWR*.

A quick glance at the keyboard will soon show you where 'E' is situated and, if one is in a hurry, distracted, or in a bit of a day dream, it is *very* easy to clip the 'E' and press Return before realising it. The next time you try to access or modify any files on that disc you will not be able to, because the ADFS system has set the 'E' attribute, which means that many years work of valuable data is lost forever.

That is what happened to me recently. I immediately questioned the statement that once set, it was impossible to remove, and with the help of PRES (John Huddleston) I was able to remove the dreaded 'E' and regain the years of collected data that I thought lost forever.

Why didn't you keep backup-copies of such valuable discs you ask. I do. I did. I run 3 identical discs, but in my day-dreaming haze I had subconsciously gone through the identical *ACC. * LWR without looking at the screen and had lost my two backups as well

This is when panic set in and that is why I feel the following information is invaluable and needs to be made available to all, because I'm sure I can't be the only person to suffer in this way!

FINDING THE 'E' ATTRIBUTE

It is essential that you have a disc sector editor to perform the following

operations. I use the *ADT* (Now PRES) *Advanced Disc Toolkit*, although a similar routine could be followed by users of other disc editors

Let's start by having a look at how the directory information is organised

Insert a disc into drive 0 and type **DEX 0*. The information on all files in the $ (root) directory is found somewhere between the second and sixth sectors. Use *Shift-Right Cursor* to advance a sector at a time until you see all the file names and sub-directories listed in the right-hand window. Normally sector 2 holds information on filenames and sub-directories. Look at bit 7 for the first few bytes of a filename: if bit 7 is set in byte 1 then the R attribute is set, in byte 2, the W attribute, and byte 3, the L attribute. Byte 4 is more complex; if bit 7 is set in a high hex number like D7 then the filename is another directory. If it is in a low hex number like 54, then the filename is a file. Finally, if the fifth byte has bit 7 set, then the E attribute is set. The setting or unsetting is done by adding or subtracting 128 decimal (&80 hex) to the value of the hex number shown in the left hand window when the cursor is positioned under the 5th character of the filename in the right hand window. Let's look at it in practice.

SETTING THE 'E' ATTRIBUTE

For the first example, create a file called *Test3* (its contents don't matter).

1. **Position the cursor under the 5th character of the filename, in this example the 3.**
2. **Look at the left hand window where the equivalent brackets () are positioned, to find the hex number 33.**
3. **Calculate the hex value of &33+&80 (if necessary exit to Basic first and PRINT ~&33+&80). Answer=B3.**
4. **Return to the 5th character of the filename.**
5. **Press the Copy key. The () brackets will change to [] brackets.**
6. **Type B3.**
7. **Press Escape and type 'Y' to re-save the sector.**
8. **The 'E' attribute is now SET.**

To check that you have the right answer (in step 6 above), enter mode 7 and type *PRINT CHR$ &B3*. This should give the answer 3, which is the 5th character of the filename. This technique can be used to check the rest of the examples; just change &B3 to the value in step 6 in each example.

The second example looks at a longer filename,*TestEd2*, which has to be dealt with in a slightly different way.

1. **Position the cursor under the 5th character of the filename, in this example the E.**
2. **Look at the left hand window where the equivalent brackets () are positioned, to find the hex number 45.**
3. **Calculate the hex value of &45+&80 (if necessary exit to Basic first and PRINT ~&45+&80). Answer=C5.**
4. **Return to the 5th character of the filename.**
5. **Press the Copy key. The () brackets will change to [] brackets.**
6. **Type C5.**
7. **Complete as above.**

REMOVING THE 'E' ATTRIBUTE

From the above examples on setting this supposedly 'unremovable' attribute, you will probably have realised that it is actually very easy to remove. Simply modify step 3 to subtract &80 from the value obtained in step 2 instead of adding it.

(UN)LOCKING FILES IN SUB DIRECTORIES

The above examples have only dealt with files that exist in the root directory. To modify files in other directories you will need to know the sector address of the directory involved. This is exceptionally easy to find out - just type **INFO **. The address should look something like *00000F* or *000372*. So type **DEX F* or **DEX 372* to take you straight to the correct sector.

All you need to do now is to repeat the steps outlined for setting or removing the 'E' attribute

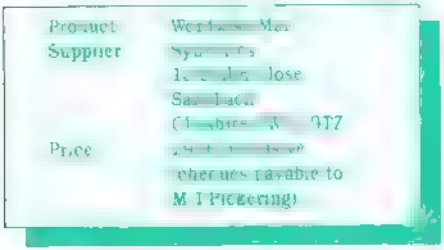
If this proves to be helpful to others then I'm happy to have provided such a service.

If other readers can unlock further dark secrets of the Beeb then perhaps they will be encouraged to submit them for publication in BEEBUG.

```
2210 PROCgetchar(F%(b,b1),b,b1)
2220 NEXT,
2230 ENDPROC
2240 :
2250 DEF PROCfsq
2260 FORqw=0TOr:FORqw1=0TOc
2270 FORqw%=0TO7:FORqw1%=7 TO 0 STEP-1
2280 IF (q%(8*qw+qw%,qw1) AND (2^qw1%))
=2^qw1% PROCfill(S%*(8*qw1+7-qw1%),S%*(8
*(r+1-qw)-qw%),1,0)
2290 NEXT,,
2300 ENDPROC
2310 :
2320 DEF PROCinst
2330 COLOUR2:PRINTTAB(30,4)"Keys :"
2340 VDU19,3,6;0;0;:COLOUR3
2350 PRINTTAB(26,6)"Arrow keys"
2360 PRINTTAB(26,7)"move cursor"
2370 PRINTTAB(26,9)"<RET> - fill"
2380 PRINTTAB(26,11)"<DEL> - blank"
2390 PRINTTAB(26,13)"S - save"
2400 PRINTTAB(26,17)"V - list VDU"
2410 PRINTTAB(26,15)"E - end"
2420 PRINTTAB,30,18)"values of"
2430 PRINTTAB(30,19)"current"
2440 PRINTTAB(30,20)"character"
2450 PRINTTAB(26,22)"You are"
2460 PRINTTAB(26,23)"editing :"
2470 ENDPROC
2480 :
2490 DEF PROCsave
2500 VDU4
2510 INPUTTAB(1,3)"Filename :- ";file$
2520 PRINTTAB(1,3)SPC(20)
2530 OSCLI("SAVE "+file$+" C00 D00")
2540 VDU5
2550 ENDPROC
2560 :
2570 DEF PROCerror
2580 VDU22,7
2590 REPORT:PRINT" at line ";ERL
2600 OSCLI("FX4")
2610 ENDPROC
2620 :
2630 DATA800,800,100,800,400,50,795
2640 DATA267,33,400,800,50,800,800
2650 DATA50,795,529,33,267,795,33
2660 DATA529,795,33,795,795,33
```

Wordwise-Mail Reviewed

Chris Robbins looks at the latest extension to Wordwise Plus, Wordwise-Mail.



Since Wordwise Plus appeared some ten years ago, there have been numerous attempts at realising the potential of its built-in programming language. Wordwise-Mail is the latest

At just under ten pounds, it's an extremely low cost product that, according to Synectics, "will help you get ... letters written more quickly, consistently and methodically than ever before!".

A better description might be that of correspondence organiser, since the major advantage over straight Wordwise Plus for writing letters, is the facilities it provides for keeping track of the letters you've written; if you use it sensibly and carefully, you need never lose any letter ever again. At least, not those you've written via Wordwise-Mail!

In order to do this it maintains a 'database' of names and addresses; an 'index' of names which allows the appropriate address to be found even if you've only got the haziest of ideas as to the correct name; and perhaps the most useful feature of all, a reference list which records identification data such as a unique program generated reference number, addressee's name, date, and a short comment that could be used later to retrieve the letter e.g. 'Scouts Outing/Luna Base II'

Getting going should be simple - well it would have been for me if I hadn't chosen to blindly follow the setup procedure described in the documentation that accompanied the review copy. I was puzzled for some time by the error message "Channel" that came up every time I tried to use it. It eventually dawned on me that the demonstration address file *!adrs* was missing from the review copy; not a very promising start.

The necessity of setting up such a file - containing the names and addresses you're going to use - isn't made clear until after the instructions to start a letter are given. In fact the documentation is generally rather lacking when it comes to describing the important, and finer, points of the program's workings

For instance, when starting a new letter, where the address isn't already in the address file, one might suppose that having entered the new address, the end of that address should be indicated in much the same way as when setting-up the original address file i.e. a circumflex ('^') on a line by itself. Not so! An investigation of the program code revealed that any line consisting of two characters or less would do, but, that they'd also become part of the address! Thus, not only would the address be wrong, but a further problem would arise (and the program crash) if '^' had been used. The correct thing to do was simply to press Return.

A more informative user prompt, and some suitable validation of the input, could have avoided these difficulties

It also suffers from an apparently inconsistent use of the Escape key It is in fact entirely consistent; it's just that sometimes you can use Escape and then again sometimes you can't. Frequent use of the program will, no doubt, bring familiarity. But for infrequent letter writers, like myself, it will, also undoubtedly, remain an extremely annoying feature.

The temptation under such circumstances is to press Break; not always a wise move! Especially if Break hasn't been 'taken in hand' by the program in use, as it hasn't in the case of Wordwise-Mail. I must admit though, that despite my finger straying onto Break on several occasions, nothing untoward occurred.

However, I found the safest and surest way to get out of trouble was to press Shift and f0 to call up the Wordwise-Mail main menu

This menu allows selection from a comprehensive set of operations that includes.

> ***starting, saving and printing letters,***
> ***setting disc drive options,***
> ***changing the default date (even for Master users this has to be done manually),***
> ***searching for letter references,***
> ***creating an index etc.***

Once in the main menu, things are fairly straightforward.

Ignoring the minor niggles of misleading documentation, the program's crashability, a less than obvious user interface, and the fact that it won't run with any version of WW+ earlier than 1.4F, the most serious drawback is that at present it only works with DFS and requires a minimum of two disc sides, one side being reserved for names and addresses. Thus the number of letters that can be saved is severely limited

For those users with only the standard DFS and just one double-sided drive the limit is 25 letters An additional double-sided drive will allow an extra 62 files, and if you've also got a utility such as DiscDoctor (which allows an extra catalogue), or maybe a DFS with a dual catalogue capability such as that from Watford Electronics, the limit can be extended quite significantly.

An ADFS version of the program would be an even better solution!

It's worth emphasizing what I said earlier about using it "sensibly and carefully". For despite claims that "You don't have to be a computer expert to use Wordwise-Mail" expertise in the WW+ programming language would come in very handy when the unexpected occurs.

On second thoughts, since the program's code is easily read, such expertise might best be employed pre-emptively, to eliminate many of the 'rough edges' of the program and make it more robust and amenable to casual use.

To sum up, the idea is a good one, and although it does come a little late in the life of the BBC Micro/Master and Wordwise Plus, the facilities provided by Wordwise-Mail, in terms of code per pound, are excellent value for money. However, more needs to be done to increase its robustness, and to improve both the user interface and the accompanying documentation.

Quasimodo

by Jonathan Temple

Quasimodo's sweetheart, Princess Esmeralda, has been imprisoned in the wicked Baron's fortress. Can you, Quasimodo the hunchback, save her from the Baron's evil clutches?

A quick Tarzan impersonation

Before being re-united with your love you have to tackle eleven screens, but what with those cruel guards throwing rocks and arrows at you and those tricky swinging ropes it's going to be difficult

You guide Quasimodo using the Z and X keys for left and right, and Return to jump. You can 'freeze' the game by pressing Ctrl. Pressing Shift will restart it again

Quasimodo has the usual computer character's quota of three lives, one of which he will lose each time he misjudges a jump or is hit by a rock or arrow - all common occurrences when you first begin to play.

To complete a screen, Quasimodo must jump up to the bell rope and ring the bell. He is then awarded a bonus, the size of which depends on the current screen and how long it took him to complete it After the 11th screen Quasimodo gets to meet his princess for one brief moment (Ahhh...) and then he's whisked away back to the start to try again

A close shave for our hunched friend

Entering the program is straightforward enough - just type it in and away you go. Model B users without sideways RAM, however, will need to miss out any unnecessary spaces and the instructions, or set PAGE to &1200 (type PAGE=&1200) before loading the program - if you choose the latter course remember not to press Break as this will corrupt the program

The Bells!

PROGRAM NOTES

The data for the eleven screens is held in lines 2840-2890, with four numbers for each screen. The first number represents the type of screen where:

0 - is a flat wall;
1 - represents turrets;
2 - is a pit with a rope and
3 - is a pit with platforms.

The next three numbers represent the chance of an arrow, rope and boulder appearing respectively. If 0, it will not appear on that screen.

Close box not close enough Quasi

The chance is decided using RND(<number in data>). If this is equal to 1 a new arrow or boulder is made to appear. So that the arrows and boulders come at the same time and position at each go, making it easier to plan a route through each screen, the RND function is seeded using RND(-<number>). This means that the numbers produced will be the same every game.

However, if Quasimodo should reach Esmeralda, when he starts again the seed (the variable RS%) for RND is changed, which means that in some of the screens the player will have to learn a new route.

Jumping is achieved by having two arrays, A%(6) and B%(6), the data for which is in lines 2820-2830. A variable, J%, is decreased as Quasimodo leaps through the air, and this is used to obtain two values from these arrays which are then added to Quasimodo's X and Y co-ordinates. By changing the values in these two lines it would be possible to get Quasimodo to jump further - useful for cheats!

A useful procedure included in the program is PROCtriple, which when called with PROCtriple(X,Y,C,A$) will print the string A$ at tab position X,Y in colour C in triple-height characters. local variables are used for this procedure, so it can be lifted straight out and used in your own programs

The chiming sound used when Quasimodo rings a bell is taken from Ian Waugh's excellent series of articles, 'Making Music on the Beeb' (BEEBUG Vol.3 No.8 to Vol.4 No.2).

The game Quasimodo was first published in BEEBUG Vol. 4 No 7

```
  10 REM Program Quasimodo
  20 REM Version B1.4
  30 REM Author  J. Temple
  40 REM BEEBUG  October 1993
  50 REM Program subject to copyright
  60 :
  70 *TV 255
 110 ON ERROR GOTO 2430 MODE 1
 130 PROCinst PROCchars PROCinit
 160 REPEAT.L%=3 S%=0 F%=1 RS%=3
 180 REPEAT.MODE 2 PROCscreen
 210 REPEAT PROCman:IF F% PROCrope
 240 IF P% PROCarrow
 250 IF M% PROCboulder
 260 UNTIL E%
 270 IF E%=1 L% L%-1:SOUND 0,1,50,1 ELS
E PROCbonus
 280 UNTIL L%=0:IF L%=0 PROCkilled
 300 UNTIL FALSE
 310
1000 DEFPROCman
1010 Z%=Z%-5:IX=4,31,6,.
1020 IF Z%>1 PRINT ,Z% "   ",
1030 ZD=S%-X% B%=Y% C%=V% D%=W%
1040 IF INKEY 2 REPEAT UNTIL INKEY 1
1050 IF INKEY-74 IF J%+JF%=0 J%=6:N%=(I
```

```
NKEY 98 -(INKEY 67):SOUND 1,1,10,5
 1060 JF%-0:IF J% PROCjump:ENDPROC
 1070 IF F% IF X%-G%-32 GOTO 1110
 1080 IF INKEY 98 IFX%>0 X%-X% 32.W%=W%
EOR 3:SOUND 18,-10,50,1:IF V%<>231 V%-23
1:W%-233
 1090 IF INKEY 67 IFX%<1216 X%-X%+32:W%
W% EOR 7:SOUND 18,-10,50,1:IF V%<>232 V%
-232.W% 235
 1100 IF POINT(X%,Y%-64)=0 IF POINT(X%+5
6,Y%-64)=0 E%=1:F%=0
 1110 IF D%<>W% GCOL3,6:MOVE A%,B%:VDU C
%,10,8,D%:MOVE X%,Y%:VDU V%,10,8,W%
 1120 IF D%-W% FOR N=1 TO 30:NEXT
 1130 ENDPROC
 1140 :
 1150 DEFPROCjump
 1160 X%=X%+A%(J%)*N%:Y%=Y%+B%(J%)
 1170 IF X%=0 IF N%=-1 N%=0
 1180 IF X%=1216 IF N%=1 N%=0
 1190 J%=J%-1:GCOL 3,6.MOVE A%,B%
 1200 VDU C%,10,8,D%:MOVE X%,Y%
 1210 VDU V%,10,8,W%:IF J%=0 JF%=1
 1220 IF X%=1216 IF Y%=668 E%=2
 1230 ENDPROC
 1240 :
 1250 DEFPROCrope
 1260 GCOL 3,7:MOVE 640,896:DRAW G%,604
 1270 G%=G%+H%:IF G%=320 OR G%=960 H%=-H
%
 1280 GCOL 3,7:MOVE 640,896:DRAW G%,604
 1290 IF J%=0 IF ABS(G%-X%)<65 IF ABS(60
4-Y%)<65 GCOL3,6:MOVE X%,Y%:VDU V%,10,8,
W%:X%=G%-32:MOVE X%,Y%:VDU V%,10,8,W%
 1300 ENDPROC
 1310 :
 1320 DEFPROCarrow
 1330 IF R%=0 GOTO 1400
 1340 IF ABS(Q% X%)<33 IF ABS(604-Y%)<33
E%=1
 1350 GCOL 3,3:MOVE Q%,604:VDU 226
 1360 Q%=Q% 32:MOVE Q%,604:VDU 226
 1370 IF Q%<-32 R%-0
 1380 IF ABS(Q%-X%)<33 IF ABS(604-Y%)<33
E%-1
 1390 ENDPROC
 1400 IF RND(P%)-1 Q%-1216:R% 1.GCOL 3,3
:MOVE 1216,604:VDU 226
 1410 ENDPROC
 1420 :
 1430 DEFPROCboulder
 1440 IF U%-0 GOTO 1510
 1450 IF ABS(T%-X%)<33 IF ABS(672-Y%)<33
E%-1
 1460 GCOL 3,6:MOVE T%,672:VDU 227
 1470 T% T%+64:MOVE T%,672:VDU 227
 1480 IF T%>1216 U%-0
 1490 IF ABS(T% X%)<33 IF ABS(672 Y%)<33
E%=1
 1500 ENDPROC
 1510 IF RND(M%)=1 T%=0:U%=1:GCOL 3,6:MO
VE 0,672.VDU 227
 1520 ENDPROC
 1530 :
 1540 DEFPROCkilled
 1550 VDU 4,28,2,26,17,20,12
 1560 PROCtriple(3,1,2,"GAME OVER")
 1570 PRINTTAB(1,5)"Press Spacebar"
 1580 REPEAT UNTIL GET=32.ENDPROC
 1600 :
 1610 DEFPROCbonus
 1620 SOUND 2,2,81,16:SOUND 2,2,81,16
 1630 FOR N=1 TO 300:NEXT:IF Z%<0 Z%=0
 1640 VDU 4,28,2,26,17,20,12
 1650 K%=K%+1:S%=S%+Z%
 1660 IF K%=12 PROCcongrats.ENDPROC
 1670 PROCtriple(2,1,3,"BONUS = "+STR$(Z
%))
 1680 TIME=0:REPEAT UNTIL TIME>200
 1690 ENDPROC
 1700 :
 1710 DEFPROCcongrats
 1720 K%=1:VDU 26,12:IF RS%=3 L%=L%+1
 1730 PROCscreen
 1740 VDU 4,28,2,26,17,16,12
 1750 PROCtriple(3,1,3,"WELL DONE !")
 1760 VDU 26,5,18,3,5,25,4,960;636;231,1
0,8,230,18,3,6
 1770 FOR X%=0 TO 864 STEP 16
 1780 MOVE X%,636:VDU V%,10,8,W%
 1790 W%-W% EOR 7:MOVE X%+16,636
 1800 VDU V%,10,8,W%.FOR N=1 TO 40
 1810 NEXT,:PLOT 69,928,616
 1820 RESTORE 2900:N=81:*FX 15,0
 1830 FOR T-1 TO 10:READ A,D:N-N+A
 1840 SOUND 1,-15,N,D:SOUND 2, 10,N+48,D
 1850 NEXT:TIME-0
 1860 REPEAT UNTIL TIME>200:VDU 4
```

```
 1870 PROCtriple(4,21,2,"Now try again")
 1880 COLOUR 5:PRINTTAB(7,25)"<SPACE>"
 1890 REPEAT UNTIL GET=32:RS%=RS%+1
 1900 ENDPROC
 1910 :
 1920 DEFPROCtriple(X,Y,C,A$)
 1930 LOCAL A%,N%,X%,Y%
 1940 X%=&70:Y%=0:A%=10:COLOUR C
 1950 FOR N%=1 TO LEN(A$)
 1960 ?&70=ASC(MID$(A$,N%)):CALL &FFF1
 1970 VDU23,253,?&71,?&71,?&71,?&72,?&72
,?&72,?&73,?&73
 1980 VDU23,254,?&73,?&74,?&74,?&74,?&75
,?&75,?&75,?&76
 1990 VDU23,255,?&76,?&76,?&77,?&77,?&77
,?&78,?&78,?&78
 2000 VDU 31,X+N%-1,Y,253,10,8,254,10,8,
255
 2010 NEXT:ENDPROC
 2030 :
 2040 DEFPROCscore(N%)
 2050 S%=S%+N%:VDU 4,17,7,31,6,1
 2060 PRINT LEFT$("00000",5-LEN(STR$(S%)
))+STR$(S%)
 2070 VDU 5.ENDPROC
 2090 :
 2100 DEFPROCscreen
 2110 VDU 4,17,1,17,135
 2120 FOR Y%=14 TO 28
 2130 PRINTTAB(0,Y%) STRING$(20,CHR$(237
+Y%MOD2))
 2140 NEXT:VDU 17,128:RESTORE 2840
 2150 FOR N%=1 TO K%
 2160 READ V%,P%,F%,M%:NEXT
 2170 IF V%=1 PROCturrets
 2180 IF V%=2 OR V%=3 PROCpit
 2190 IF V%=3 VDU 31,5,14,225,31,8,14,22
5,31,11,14,225,31,14,14,225
 2200 Z%=K%*100+400:PRINTTAB(0,0)"BONUS:
";TAB(0,1)"SCORE."
 2210 COLOUR7:PRINTTAB(6,0);Z%;TAB(2,26)
"SCREEN:";X%
 2220 COLOUR6:IFL%>1 FOR X%=15 TO L%*2+1
1 STEP 2:VDU 31,X%,0,232,10,8,235:NEXT
 2230 PROCscore(0) D=RND.-RS%,
 2240 X%=0 Y%=636 V%=232-A%=235
 2250 E%=FALSE J%=FALSE JF%=FALSE
 2260 R%=FALSE U%=FALSE G%=320 H%=32
 2270 VDU 18,0,6,25,4,1248;636;25,5,1248
;888;18,3,3,25,4,1216;928;228,10,8,229,1
8,3,6,25,4,X%;Y%;V%,10,8,W%,23;10,32;0;0
;0;
 2280 IF F% GCOL3,7:MOVE 640,896:DRAW G%
,604
 2290 ENDPROC
 2300 :
 2310 DEFPROCturrets
 2320 PRINTTAB(4,14)G$;TAB(9,14)G$;TAB(1
4,14)G$
 2330 FOR X%=376 TO 1016 STEP 320
 2340 VDU 25,4,X%,568;25,0,0;-92;25,81,-
32;0;25,0,0,60;25,81,32;32;
 2350 NEXT:ENDPROC
 2370 :
 2380 DEFPROCpit
 2390 FOR X%=3 TO 15 STEP 2
 2400 PRINTTAB(X%,14)G$;:NEXT
 2410 VDU 25,4,1080;568,25,0,0;-92;25,81
,-32;0;25,0,0;60;25,81,32;32;
 2420 ENDPROC
 2430 :
 2440 DEFPROCinit
 2450 DIM A%(6),B%(6)
 2460 FOR N%=1 TO 6
 2470 READ A%(N%),B%(N%):NEXT:RS%=3
 2480 G$=STRING$(3," "+CHR$10+CHR$8+CHR
$8)
 2490 ENDPROC
 2500 :
 2510 DEFPROCinst
 2520 VDU17,130,28,10,5,28,1,12,26
 2530 PROCtriple(11,2,1,"Q U A S I M O D
O")
 2540 VDU19,3,6;0;17,128,17,3,31,0,8
 2550 PRINT" In this version of the wel
l-known"'"arcade game you must guide Qua
simodo"'"through the eleven screens to h
is"'"sweetheart, Princess Esmeralda."'
 2560 PRINT" Our hero must avoid the ar
rows and"'"rocks the cruel guards are th
rowing at"'"him, and use the ropes to sw
ing across"'"the dangerous gaps."'
 2570 PRINT" You should use the Z and X
 keys to"'"move left and right, and <Ret
urn> to"'"jump. To complete each screen
Quasimodo"'"must jump up to the bell rop
e and ring"'"the bell."'
```

Continued on page 24

Machine Code Corner

In which Mr Toad gives us some answers.

In recent issues, you may recall, Mr T has been catching up with the correspondence; since last time I've had some nice letters - and some useful literature - from Cliff Blake of Portsmouth and Bill Woodall of Yeovil. Sorry guys, we'll have to wait until a later issue to deal with your excellent contributions, because this month it's the turn of Arthur Adams Arthur sent me some handy references, and also said nice things about me in the July issue, for which he deserves an *I'M A SWOT* badge. Arthur pointed out that I was being too clever in my full-sized ROM header code (BEEBUG Vol 11 No.8). I used a variable Z% to hold the initial value of O%, the start of the assembly area, determining the lowest possible address by:

```
240 Z%=?2+&100*?3
```

BUT.....

It doesn't always work, and it's worth telling you why, since we end up with a useful way of saving memory when space is tight. Addresses 2 and 3 hold *V-TOP*, the address of the first byte above the Basic variables. However, at the start of the first pass of the assembler there aren't yet any variables, so Z% points to the first location after the Basic assembler program. As the first pass proceeds, the assembled code is overwritten by the growing variables area, but that's OK. At the beginning of the second pass, addresses 2 and 3, having been updated, now point to the new *V-TOP*, so assembly proceeds from there, the first free byte above the variables. So what's wrong with DIM statement? OK, it starts the assembly off at the same place as our trick, but what about the top end? You have to make a guess as to the space you'll need You can run the program to find out, then alter the DIM, but that's more trouble than my way, let alone the fact that it would need revision after any alterations.

All fiendishly clever stuff, and I'd never had any trouble with it before, but in Arthur's program there must have been a large number of variables or labels declared in one short stretch. At some point during the first pass, some new variable, as it was declared, went into the space above the the assembled code. Later the code being assembled caught up again and overwrote that variable, so when the machine next looked for it, it wasn't there. At this point the Beeb sensibly stopped assembling and gave a rude message.

Sod's Law - the first time my cunning scheme gave trouble was *after* publication! I hadn't thought deeply enough, and had reasoned that it couldn't happen. There's a moral in there somewhere. The fix is to leave some just-in-case space on the first pass by starting assembly a good way *above TOP/V-TOP*, then to bring the starting-point down to *V-TOP* for the second pass.

The two passes of the assembler are done by:

```
FOR N% = 4 TO 6 STEP 2:OPT N%
```

When we set Z%, we can test *N*% to find out which pass is being executed The statement *N*%=4 will evaluate to -1 if TRUE, or 0 if FALSE; thus we could multiply *N*%=4 by the amount of spare space we mean to allow. Say we decide on &200 bytes: &200*(*N*%=4) will evaluate to -&200 on the first pass (-1*&200), but to 0 on the second, since 0*&200=0. An ABS will remove the minus: ABS(&200*(N%=4)) now equals

&200 on the first pass. Bingo! The line can now read

```
240 Z%=?2+&100*?3+ABS(&200*(N%-4))
```

Assembly of the object-code now wastes not a single byte. Believe me, it can matter if the program is fairly long. But, gentle reader, we can simplify the line a bit. That's this month's competition, part one: remove the ABS() from line 240 and make just *one* other change so that it works exactly as before

Just one teensy-weensy snag - you may get into trouble if you declare another variable later in the program. The only reason I can think of for wanting to do so in the assembly text of a ROM is a 'find-a-free-slot-for-the-*SRWRITE' loop such as I used in the published headers; there I used a resident integer, *N%*, which lives in a different area of memory. This sparks an idea - if you're finishing a project in assembler and you're really short of space, why not use *A%* to *Z%* to replace some of the variables? You'll gain a lot of space. You could also use them as labels - and this brings us to part two of this month's competition: although you can use resident integers as labels, you'll have to do three passes of the assembler or run the program twice before saving the code. Why? Do write in: there are more than enough badges left and Mr T loves to hear from readers

Anyhow, as promised in the last issue, (which was on machine-code arithmetic, remember?) we'll now have a look at another way in which the 6502 allows us to manipulate a byte, other than by arithmetic.

This month we'll look at the 'rotate' or 'shift' operations, which mean moving each bit of a byte one place to the left or right. This is handy for multiplying or dividing by 2 (or 4, 8, 16, etc. if you repeat the operation) We saw this last time.

All four of the rotate/shift instructions can be used with five addressing modes:

Accumulator	write A after the instruction to rotate *A*
Zero-page	write a one-byte address *nn* after the instruction
Zero-page,X	write *nn*,X after the instruction
Absolute	write a two-byte address *nnnn* after the instruction
Absolute,X	write *nnnn*,X after the instruction

To multiply a byte, rotate it to the left by ASL - Arithmetic (*NOT* 'Accumulator') Shift Left, or ROL - ROtate Left. The difference between the two is that after a shift, the least significant bit of the byte, bit 0, always ends up clear (=0), whereas after a rotate, bit 0 holds what was previously in the carry-flag. Both finish by moving the bit which 'falls off the end' into the carry-flag. One reason for having the two is that you can multiply a multi-byte number, using the carry-flag to move bit 7 of one byte into bit 0 of the next. Let's multiply by two a two-byte number held LSB-MSB at *.number* - but, whatever method we use to double an n-byte number, we must have n+1 bytes to store the result, in case there's an overflow.

```
LDA #0
STA number+2   \ ready to store any
                 overflow at end.
ASL number     \ we want bit 0 of the
                 low-byte clear, so not
                 ROL. C flag now =
                 original bit 7 of number.
ROL number+1   \ contents of &71 now *2 +
                 original bit 7 of number.
                 C flag now = previous bit
                 7 of number+1.
ROL number+2   \ Rotate C flag into bit 0
                 of number+2.
```

It's just like including the carry-flag in an addition or subtraction, only with those operations, the 6502 doesn't have carry-flag-excluded versions, so we have to use CLC or SEC in preparation. Happily, here we have both rotate and shift available

Another reason for having the rotate instruction is that, if you keep repeating it on a single byte, the byte literally rotates - thanks to the carry-flag, the bits which are rotated off the high end get put back into the low end Thus, if you want to multiply a *single* byte, you must use ASL, not ROR, otherwise you are likely to get some bits added over and above the simple multiply.

You can also shift and rotate to the right, using LSR or ROR. That's to divide, of course All the above remarks apply as regards the carry flag.

You may sometimes perform one of these operations for reasons other than arithmetic. Fancy screen dissolves, for example Most of the fonts in my *Fontz ROM* (BEEBUG Vol.10 No.1) were achieved by manipulating the bit-patterns of the characters, using ASLs or LSRs plus the operations which we shall consider next month.

That's it for now, reptile-readers. Another notch carved on Mr T's monitor. I'm off to the sweet shop to spend this month's money on a sherbet dip - I bet I can swell up bigger than those fancy South American frogs

If I go pop, think only this of me.
There is some corner of a Beebug desk
That is forever reptile

Next month: AND, OR and EOR (no, not Christopher Robin's donkey!). Also, who is the Patron Saint of computing, and why not?

Quasimodo (continued from page 21)

```
 2580 PRINT"  <Ctrl> can be used to free
ze the game""until <Shift> is pressed."

 2590 COLOUR2:PRINT'TAB(5)"Press the SPA
CE BAR to play..."
 2600 REPEAT UNTIL GET=32:ENDPROC
 2620 :
 2630 DEFPROCchars
 2640 VDU23,225,-1,-1,-1,239,193,0,0,0
 2650 VDU23,226,33,66,-1,66,33,0,0,0
 2660 VDU23,227,60,94,182,175,187,183,94
,60
 2670 VDU23,228,24,36,24,44,44,44,94,94
 2680 VDU23,229,94,-1,129,126,0,0,0,0
 2690 VDU23,230,60,172,92,24,56,60,62,12
6
 2700 VDU23,231,56,124,76,38,194,70,60,2
4
 2710 VDU23,232,28,62,50,100,67,98,60,24
 2720 VDU23,233,60,118,118,110,60,24,24,
56
 2730 VDU23,234,60,110,118,118,118,60,24,60,
100
 2740 VDU23,235,60,110,110,118,60,24,24,
28
 2750 VDU23,236,60,110,118,118,60,24,60,
38
 2760 VDU23,237,-3,-3,-3,-3,-3,-3,-3,0
 2770 VDU23,238,223,223,223,223,223,223,
223,0
 2780 ENVELOPE 1,133,8,4,8,3,1,1,126,0,0
,-10,126,0
 2790 ENVELOPE 2,4,0,0,0,0,0,0,126,-1,-1
,-1,80,0
 2800 ENDPROC
 2810 :
 2820 DATA 0,-16,32,-16,32,-16,32,16,32
 2830 DATA 16,0,16,0,2,0,0,1,0,0,0,2,0,1
 2840 DATA 0,3,0,0,0,0,2,0,4,1,2,0,0,2,9
9
 2850 DATA 1,0,3,0,0,2,1,5,0,2,2,99,1,2
 2890 DATA 3,10,0,2,0,4,8,4,8,4,4,4,8,8
 2900 DATA  12,8,4,8,  12,8,8,8,  16,8
 2920 :
 2930 MODE7:PRINT'':REPORT
 2940 PRINT " at line ";ERL.END
```

Error Handling

Alan Wrigley describes how to take advantage of Basic's reporting of errors.

Sooner or later, all programmers need an understanding of the way in which errors are handled and reported in Basic. Errors can occur for all sorts of reasons, and few programs are totally immune to them. Errors could be caused, among other things, by mistakes in programming, such as typing errors in keywords, by faulty logic, or by failing to predict an unusual sequence of actions by the user. There are many other causes of errors, and the larger and more complex the program, the greater the chances that errors will occur. With a substantial program it often requires more time to sort out errors than to code the program in the first place.

Basic makes a distinction between two different types of error: fatal and non-fatal errors. Fatal errors are those which cause the program to terminate automatically without further ado; an example would be "No room", which clearly must be fatal since the program cannot run if there is insufficient memory. Non-fatal errors, on the other hand, can be *trapped* by the program, which means that they can either be ignored, or reported in some way without stopping the program. Most of Basic's errors are non-fatal, and I will be describing how to trap them later.

For programming purposes, we can divide non-fatal errors further into serious and not-so-serious errors. For example, if a call is made to a non-existent procedure, a "No such FN/PROC" error will be generated This could be trapped, but since the program is unlikely to be able to continue sensibly with a procedure missing, the error can be considered as serious enough to terminate the program. On the other hand, if the user is asked to give a filename, the program should not just give up the ghost if the file can't be found on the current disc - perhaps the name has been typed incorrectly or the wrong disc is in the drive.

A good programmer will aim to eliminate *all* potential serious errors before the program is to be used. In the ideal world, no program should crash under any circumstances; if a problem arises it should cope with it in a dignified fashion, issuing a warning to the user so that adjustments can be made to input etc.

Errors in programming or faulty logic usually show themselves up at an early stage provided that programs are tested thoroughly while in the process of being written. However, trying to cater for all possible environments or all possible actions by the user (some of which may be extremely silly but you have to handle them all the same) can be very difficult, and is the main cause of most errors. Take the example given above, where the user is asked for a filename. You may assume that ADFS is the current filing system, and allow 10 characters for the filename But the user may have switched to DFS before running the program, which will result in a crash if you try to save a file with a name of more than 7 characters

As another example, you might be using EVAL to evaluate an expression typed in by the user. However, any attempt to use EVAL on an expression which does not make sense to Basic will generate a "No such variable" error. This includes null strings (as would result if just Return was pressed and nothing else). You must be prepared for this and either code your program in such a way that an error is impossible, or at the very least trap the error if it occurs.

ERROR REPORTING

Basic has a number of keywords associated with errors The REPORT

statement can be used to display the message relating to the last error that was generated (e.g. "No such variable", "Escape" and so on), regardless of whether that error was generated from within a program or at the Basic command line If no error has occurred since the computer was last switched on or reset, a simple copyright message is displayed. This statement can be used within error trapping routines, for example to display the error message while at the same time preventing the error from reaching the outside world and terminating the program.

Two other useful related keywords are ERR and ERL, which are functions returning respectively the error number of the last error, and the program line at which it occurred. Both of these can be useful in an error-handling routine - the former to help you decide which errors to trap and which to pass on, and the latter in conjunction with REPORT to indicate where the error occurred. In general, the line number is only useful while debugging a program, since it is normally unnecessary (and undesirable) for the user to be given such information

TRAPPING ERRORS

Normally when an error occurs in a Basic program, Basic stops execution immediately and displays the appropriate error message suffixed by the phrase "at line n", where n is the line number at which the error occurred. The value of ERL is set to this line number, while the value of ERR is set to the error number. Every error has its own number so that each can be identified Basic uses numbers from 0 to 45, while other ROMs such as the MOS and filing systems use other numbers up to 255. A list of Basic's error numbers is given in the User Guide and in the Master Reference Manual. Other error numbers may be described in literature relating to the ROM in question; if all else fails you can find out the number of any error by forcing it to occur and then reading the value of ERR immediately afterwards.

In order to stop this process happening automatically, Basic provides a means for trapping non-fatal errors This is done by using the ON ERROR statement. This tells Basic that when an error occurs, it must execute the statement immediately following the ON ERROR. Some examples of its use might be as follows:

```
ON ERROR GOTO 5000
ON ERROR PROChandle_error
```

In the third example, the error handler is contained in the rest of the line (in this case simply duplicating Basic's default action by reporting the error and then quitting). In the other two examples, a procedure or routine is pointed to. This routine can perform whatever actions you require on encountering an error. Often it will involve looking out for certain likely errors (such as the non-existent filename mentioned above) and giving some kind of warning, while either ignoring other errors or simply reporting them. In many cases, in fact, "error-trapping" is something of a misnomer since it is really a question of providing feedback to which the program can react.

A very important point to note is that after executing the remainder of the ON ERROR line, Basic continues from the line following the ON ERROR. In other words, irrespective of where in the program the error occurs, trapping it with ON ERROR will always return control to the point in the program after the ON ERROR Not only this, but Basic re-initialises its stack, which means that all loops, procedures etc. are closed, and so effectively it forgets where it was before the error. This has important implications for the way you program error handlers.

With a structured approach to program design, this problem need not bother you too greatly. If you have a main program loop, which perhaps puts a main menu on the screen and then calls one of a number of procedures depending on the user's choice, then you could put your ON ERROR statement immediately

before the main loop and be sure that any errors will return to a well-defined point

It is also possible to have more than one ON ERROR statement in a program. The most recent statement encountered will be the one which is executed in the case of an error. This means that you could invoke a different error-handler in one section of the program if you wish. However, you then have to be careful about your structure - Basic will return to the point following the new ON ERROR, and if this is in a procedure this fact will have been forgotten. When the ENDPROC is encountered, a "No proc" error will be generated, invoking the error handler again and thus producing an infinite loop.

Probably the safest way of overcoming this is to stick with the main error handler, but to set a flag depending on where the error is called from. For example, you could set a variable *err%* to a different value at the start of each procedure, and reset it to zero at the end of the procedure. Next time the program returns to the start of the main loop, if *err%* is non-zero this means that an error must have occurred in the relevant procedure, and it can be called again straight away.

To finish the article, let's look at a concrete example of an error handler. We will assume that all serious programming faults have been removed, and any errors generated by the program will be down to incorrect user input. The two errors we have decided to trap are "Escape", which occurs when Escape is pressed, and "Not found", which will occur if a file is not found. In the first case we want to return to the start of the main program loop whenever Escape is pressed, while in the second case we want to warn the user that the filename typed in doesn't exist. Other errors we will simply ignore for now.

Before the main loop, therefore, we need a couple of lines such as:

```
ON ERROR PROChandle_error
IF err%-1 THEN PROCinput ELSE .
```

The list of procedures in the second line will depend on your program; for the purpose of our example we have assumed that if the "Not found" error is going to occur, it will be in *PROCinput*, and this procedure sets *err%* to 1. The error handling procedure itself will look something like this:

```
DEF PROChandle_error
IF ERR 17 err% : ENDPROC
[illegible] ELSE
REPORT PRINT" at line "ERL
PRINT"Press a key to continue"
REPEAT UNTIL GET
ENDPROC
```

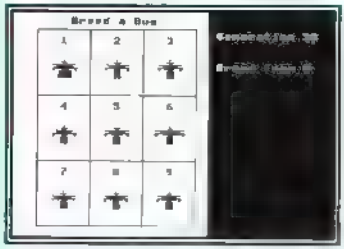
Error number 17 is "Escape"; if this occurs we must set *err%* to zero since we actually want the program to run through the main loop again rather than return to where Escape was pressed. "Not found" is error number 214; here we have printed a very simple message, but you could do something more elaborate if desired. All other errors are merely reported, and finally the procedure prompts for a keypress and waits until it has been made.

In a complex program, of course, error handling will probably need to be much more elaborate. However, this simple example gives you some idea of the techniques involved.

One final point that really needs to be made is that your program should wherever possible be written in such a way that errors do not occur in the first place - in other words you should not just rely on Basic to report things which could have been avoided. An example might be where the user has to type in some input; if you check what is entered (or in the case of numbers, say, only allow numerals to be entered in the first place), you will avoid errors later such as the problem with EVAL mentioned earlier. You should always aim to pre-empt errors in this way and treat the error handler as a last resort. BJ

Best of Acornsoft

Applications I Disc



Business Graphics for producing graphs, charts and diagrams
Video Cataloguer - catalogue and print labels for your video cassettes
Phone Book an on screen telephone book which can be easily edited and updated
Personalised Letter-Headings design a stylish logo for your letter heads
Appointments Diary a computerised appointments diary
Mapping the British Isles draw a map of the British Isles at any size
Selective Breeding a superb graphical display of selective breeding of insects
The Earth from Space - draw a picture of the Earth as seen from any point in space
Personalised Address Book - on-screen address and phone book
Page Designer a page-making package for Epson compatible printers
World by Night and Day a display of the world showing night and day for any time and date of the year

File Handling for All

on the BBC Micro and Acorn Archimedes

by David Spencer and Mike Williams

Computers are often used for file handling applications yet this is a subject which computer users find difficult when it comes to developing their own programs. *File Handling for All* aims to change that by providing an extensive and comprehensive introduction to the writing of file handling programs with particular reference to Basic.

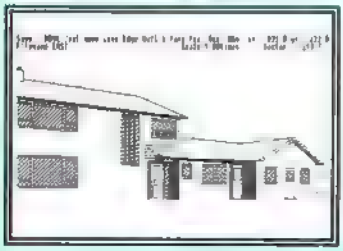
File Handling for All, written by highly experienced authors and programmers David Spencer and Mike Williams, offers 144 pages of text supported by many useful program listings. It is aimed at Basic programmers, beginners and advanced users, and anybody interested in File Handling and Databases on the Beeb and the Arc. However all the file handling concepts discussed are relevant to most computer systems, making this a suitable introduction to file handling for all.

The book starts with an introduction to the basic principles of file handling, and in the following chapters develops an in depth look at the handling of different types of files e.g. serial files, indexed files, direct access files and searching and sorting. A separate chapter is devoted to hierarchical and relational database design, and the book concludes with a chapter of practical advice on how best to develop file handling programs.

The topics covered by the book include:

Card Index Files, Serial Files, File Headers, Disc and Record Buffering, Using Pointers, Indexing Files, Searching Techniques, Hashing Functions, Sorting Methods, Testing and Debugging, Networking Conflicts, File System Calls

The associated disc contains complete working programs based on the routines described in the book and a copy of Fter, a full-feature Database program originally published in BEEBUG magazine.



ASTAAD

Enhanced ASTAAD CAD program for the Master, offering the following features:

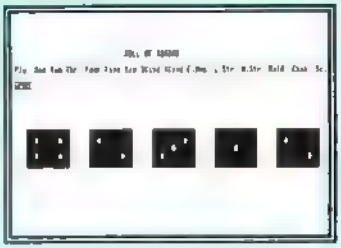
- full mouse and joystick control
- built-in printer dump
- speed improvement
- STEAMS image manipulator
- Keystrips for ASTAAD and STEAMS
- Comprehensive user guide
- Sample picture files

	Stock Code	Price		Stock Code	Price
ASTAAD (80 track DFS)	1407a	£5.95	**ASTAAD** (3.5" ADFS)	1408a	£5.95
Applications II (80 track DFS)	1411a	£4.00	**Applications II** (3.5" ADFS)	1412a	£4.00
Applications I Disc (40/80T DFS)	1404a	£4.00	**Applications I Disc** (3.5" ADFS)	1409a	£4.00
General Utilities Disc (40/80T DFS)	1405a	£4.00	**General Utilities Disc** (3.5" ADFS)	1413a	£4.00
Arcade Games (40/80 track DFS)	PAG1a	£5.95	**Arcade Games** (3.5" ADFS)	PAG2a	£5.95
Board Games (40/80 track DFS)	PBG1a	£5.95	**Board Games** (3.5" ADFS)	PBG2a	£5.95

All prices include VAT where appropriate. For p&p see Membership page.

Best of BEEBUG

Board Games



Solitaire - an elegant implementation of this ancient and fascinating one-player game, and a complete solution for those who are unable to find it for themselves.

Roll of Honour Score as many points as possible by throwing the five dice in this on-screen version of Yahtzee

Patience - a very addictive version of one of the oldest and most popular games of Patience

Elevenses - another popular version of Patience lay down cards on the table in three by three grid and start turning them over until they add up to eleven

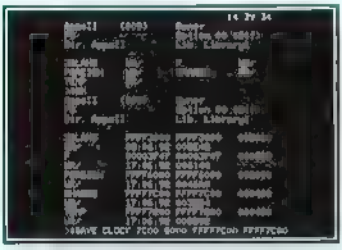
Cribbage - an authentic implementation of this very traditional card game for two, where the object is to score points for various combinations and sequences of cards.

Twiddle a close relative of Sam Lloyd's sliding block puzzle and Rubik's cube, where you have to move numbers round a grid to match a pattern

Chinese Chequers a traditional board game for two players, where the object is to move your counters following a pattern, and occupy the opponent's field.

Aces High another addictive game of Patience, where the object is to remove the cards from the table and finish with the aces at the head of each column

Applications II Disc



Crossword Editor - for designing, editing and solving crosswords

Monthly Desk Diary - a month-to-view calendar which can also be printed

3D Landscapes generates three dimensional landscapes

Real Time Clock - a real time digital alarm clock displayed on the screen

Running Four Temperatures calibrates and plots up to four temperatures

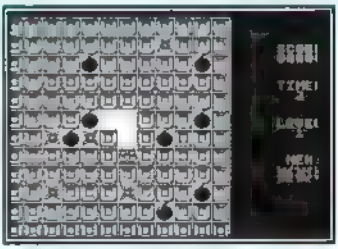
Julia Sets - fascinating extensions of the Mandelbrot set

Foreign Language Tester - foreign character definer and language tester

Share Investor - assists decision making when buying and selling shares

Label Processor for designing and printing labels on Epson compatible printers

Arcade Games



George and the Dragon Rescue Hideous Hilda from the flames of the dragon, but beware the flying arrows and the moving holes on the floor.

Ebony Castle - You, the leader of a secret band, have been captured and thrown in the dungeons of the infamous Ebony Castle. Can you escape back to the countryside, fighting off the deadly spiders on the way and collecting the keys necessary to unlock the coloured doors?

Knight Quest - You are a Knight on a quest to find the lost crown, hidden deep in the ruins of a weird castle inhabited by dangerous monsters and protected by a greedy guardian.

Pitfall Pete - Collect all the diamonds on the screen, but try not to trap yourself when you dislodge the many boulders on your way

Builder Bob - Bob is trapped on the bottom of a building that's being demolished Can you help him build his way out?

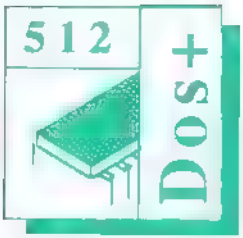
Minefield - Find your way through this grid and try to defuse the mines before they explode but beware the monsters which increasingly hinder your progress

Manic Mechanic - Try to collect all the spanners and reach the broken-down generator before the factory freezes up

Quad - You will have hours of entertainment trying to get all these different shapes to fit.

	Stock Code	Price		Stock Code	Price
File Handling for All Book	BK02b	£9.95			
File Handling for All Disc (40/80T DFS)	BK05a	£4.75	**File Handling for All** Disc (3.5" ADFS)	BK07a	£4.75
Joint Offer book and disc (40/80T DFS)	BK04b	**£11.95**	**Joint Offer book and disc** (3.5" ADFS)	BK06b	**£11.95**
Magscan (40 DFS)	0005a	£9.95	**Magscan Upgrade** (40 DFS)	0011a	£4.75
Magscan (80T DFS)	0006a	£9.95	**Magscan Upgrade** (80T DFS)	0010a	£4.75
Magscan (3.5" ADFS)	1457a	£9.95	**Magscan Upgrade** (3.5" ADFS)	1458a	£4.75

All prices include VAT where appropriate. For p&p see Membership page.



512 Forum

by Robin Burton

This month we'll start to investigate a simple-to-use back-up system based on readily available software

For my examples I'll use PKZIP, but first, thanks to Howard Smith for providing information on an early version of Flexibak and to David Harper for supplying a copy of the latest version, now called Flexibak Plus. More on these next month

GETTING ORGANISED

Whatever back-up system you use, your first thoughts should be about how you'll want to secure your files and how they might need to be recovered. You won't always have had a total disaster, so you won't always want a complete recovery, and it could even be just one file

To plan the job, mentally divide your data and programs into logical chunks, each of which can be processed separately as and when required, not just as part of the whole. I'll explain this concept in more detail later, using my own data to illustrate. Your approach will depend to an extent on the facilities in the software used, but some generalisations apply, whether you're floppy or hard disc based. That said, the two disc types also impose constraints of their own which must be considered

Hard disc users really must plan back-ups more carefully than floppy users, because no matter what tools are used there's no way the contents of even a small winchester will fit onto a single floppy.

By contrast floppy users store files on separate discs already, so back-up planning might seem unnecessary. Not so! If the crude approach of whole disc copies is your method there are gains to be made, potentially spectacular ones, when data compression is employed. Files from a particular application might fill several floppies, but the number of back-up discs can be reduced dramatically if the data is compressed. Equally important, securing and recovering can be very much quicker than direct copies, as we'll see later.

The best approach also depends on the type of file and whether new files are continually created, or more typically only existing ones are updated. If your back-up system is currently not very well organised (perhaps even if you think it is) here are some ideas to consider.

Let's illustrate extremes by assuming the system is used for only one application. If the task is producing letters and reports, it's likely new files will continually be created. If a permanent disc record is needed, each new file must be included in your back-ups. Alternatively, if the sole use of the system is for a database new files won't be created very often, perhaps never, once the system is set up. Here the need is to secure the latest versions of existing files as they're updated.

Like most users I'm somewhere between the two extremes, usually amending files, but occasionally creating new ones

too. Some compromise is inevitable, but you should still tune your data organisation and back-up strategy to suit your needs. The ideas here are food for thought and, while some points may be obvious, it's unlikely that there's no room for improvement, unless you're very well organised. To a degree hard disc users have to be more organised, while floppy users can be a bit more flexible in their arrangements, but both might benefit from a bit more thought.

REDUCING THE WORKLOAD

The first point is that efficient security means saving time, which means selective back-ups, and avoiding duplicated effort is the biggest step you can take towards speeding up the job. This will be harder if your programs and data aren't organised with this in mind in the first place.

Applications usually have a number of files, even a word processor probably has overlays, one or more dictionary files, various printer related files, plus help and configuration data too. As an example, the volume of data in my word processor is 460K contained in ten files. Although word processing is a 'simple' application (some are much worse) it's a big help not to secure such files unnecessarily.

Remember too that EXE files (256K of my total) and dictionaries (107K) are less likely to compress well, so here's almost half a megabyte of back-up which I don't repeat unless there's a good reason.

Obviously, if different file types are mixed together in the same directory selective back-ups (and recoveries) will be more difficult. If you can't separate the different types of files, you'll be continually re-securing application files that haven't changed - a valuable waste of time and disc space.

Mind you, even if you have original issue discs, don't think you needn't secure your applications. Remember, configuring some applications to your particular requirements can be quite a lengthy job, and one that's best avoided unless there's a good reason for it. If problems force you to re-copy an application you'll have to repeat the entire installation process again, unless you have a secure copy of the fully configured working system.

Applications should certainly be secured, but only once. It shouldn't be done just because you're securing data after a work session. Keep application files and data files either on separate discs, or at least in separate directories and then, with even the simplest procedures, you can secure the application once, and thereafter save time by securing only the data

This also simplifies recovery, since it allows easy recovery of data, applications, or both when necessary. Mix them together and recovering the just the data, or just the application becomes much harder, if not impossible.

MAKE LIFE EASY

The biggest step towards back-up reliability is automating the process as much as possible. Why? Two main reasons.

The first is subjective - if the system's easy to use, it will be easier to discipline yourself to use it. If it's fiddly or involves too much effort, human nature says it won't be done as often as it should be. The second reason is that during back-ups mistakes will be far less likely. A fixed routine which is tried and tested

doesn't offer much chance to introduce new problems.

Using an essentially manual system, like PKZIP, two standard facilities (ie. free and available) will help. The first is batch files, the second is on, which is frequently overlooked. Redirection can provide extra control or a range of options simply and with total reliability.

Now let's look at real examples. These are based on my routines for securing Essential Software's files. The data structure is simple, one directory called ES, in the root directory of my hard disc, contains everything belonging to ES. Source files, issue disc contents, and sundry data files are all stored in one level of sub-directories within ES I won't list the whole structure, but, part of it looks like this:

```
\ES
\ES\DEV
\ES\NEWPROGS
\ES\SOURCES
\ES\RAMDISC
\ES\PRTSCRN
\ES\SUPRSTAR
```

....and so on, totalling 22 subdirectories containing about 3.25Mb of data.

DEV and NEWPROGS are development and testing directories, so their contents tend not to remain fixed for long. SOURCES changes less often, only when a modification to an existing program is completed or if a new one is added.

Finally, from RAMDISC onwards, the ES issue directories are virtually static, the only changes being bug-fixes (none for years, of course) or if a spelling error or 'typo' is corrected in the documentation.

You can see, given that the initial back-up included everything, I need to secure DEV and NEWPROGS fairly frequently, SOURCES less often and issue directories rarely if ever. Apart from the differing frequency required by these back-ups, the total volume of ES, even compressed, is obviously far too much to fit onto a single floppy. So how do I handle it?

REDUCING RISKS

Like other systems, PKZIP can create single archives spanning multiple floppies, but it's an option I prefer not to use. Instead the ES back-up is split into three jobs, one for DEV and NEWPROGS, one for SOURCES, and one for the issue directories. You might favour multi-volume archives but I don't, here's why

First, if one disc in a multi-volume set becomes faulty, the whole set is at risk Perhaps you can re-create a faulty disc from a duplicate copy of the set, but remember that this requires an exact duplicate; nothing else will do. If each back-up disc is independent and one develops an error, the worst that can ever happen is that you lose the contents of that one disc; it can't affect the rest

Second, when you update a multi-volume set (in all the systems I know) you have to update several if not all of the back-up discs each time. At best this is likely to be the first disc (holding the index), the disc or disc's containing the files to be updated and the last in the set (or an extra disc) which is where copies of amended or new files are usually added. As mentioned, if you're wise you'll have two exact duplicate sets, so you actually have to do the whole job twice, or copy each of the amended discs every time, (but beware, if you use COPY and miss a disc, or there's an undetected error in the first set....!).

The 'independent disc' approach avoids all this. The only discs you handle are those containing amended files, and so long as your back-up software is fast enough (mine is), you simply make two originals of every back-up disc

Spanned-volume archives are inflexible and seem to offer no benefits to balance the inconvenience and potential risks. You can disagree, but I have had a case (only one admittedly, but it's enough) where both copies of a back-up disc were faulty.

PERFORMANCE

Last month I said that PKZIP commands could become complex. However, they don't have to be, so let's take a look The general format of the command is.

```
PKZIP [option] <archivename> [file-list]
```

where *[option]* can be one or several action modifiers, *<archivename>* (which can include a drive/path specification) is the output archive's filename and *[file-list]* is a list of files to be processed.

If omitted *[option]* defaults to 'add', which writes a new version of all (source) files to the archive, whether the file or the archive already exists or not - if they don't they're created. By default *[file-list]* is everything in the current directory if no list is supplied. The archive name must be specified of course, but the extension .ZIP is added automatically.

Suppose then I want to secure the entire contents of ES\SOURCES to drive A:. If my current directory is ES\SOURCES and the archive is called ES_SRCES, the command can be as simple as.

```
PKZIP A:ES_SRCES
```

Everything but the archive path and name are defaulted, so this couldn't get much easier

I also said compressed archives are much more efficient than direct copies, so by how much? ES\SOURCES contains 51 files totalling 1.628Mb. This, of course, would need at least two 800K floppies using COPY, perhaps three, but I tried it to quantify the task.

After 2 mins 55 secs the expected 'disc full' message appeared, with 29 files copied. I lost interest at that point because copying the remaining files would obviously take a great deal of manual entry, hence the next disc (or two) would take very much longer. Let's be very generous and suppose the whole job could be done in 10 minutes

Next I timed the job using PKZIP version 2.04g. The results speak for themselves. ES_SRCES.ZIP was completed in under three minutes, it contained all the files, and was less than 325K. This is a compression ratio of better than 5:1, or put another way I could get over twice as much data on the disc in less time than taking one direct copy. Of course, this data was text so compression should be good, but the comparison is valid because the same real data was used which was identical in both cases

What about a 'poor' example, so as not to be accused of bias? OK. I also archived my word processor's directory to drive A:, producing an archive file of 250K in 1 min 40 secs. The compression is very impressive and worthwhile, especially given the file types, but here the elapsed time was slightly more than for a direct copy (1 min 18 secs)

This further reinforces the earlier point, that securing applications should be avoided unless it's justified by changes in

Continued on page 38

Public Domain

Alan Blundell is pleasantly surprised by the amount of new PD software which he has come across recently.

Having only recently written in this column that the amount of new PD software appearing was in decline, this month I'm going to cover the largest set of new discs I've ever added to my library in one month!

For Electron users, I'm pleased to announce the availability of ELBUG. As I'm sure readers will know, ELBUG was the sister publication to BEEBUG, many years ago now (before RISC User had even been thought of). It ran for 22 issues before being discontinued and I have received permission to distribute all of the software discs associated with the magazine. I've prepared 3 discs (in ADFS 'M' format) which hold all 22 issues worth of software I won't go into the contents of the discs in detail here, except to say that much of the software associated with ELBUG consisted of 'Elk' versions of programs featured in early volumes of *BEEBUG*. It is however, one of the few software collections which is guaranteed to be fully Electron compatible.

Concerning the early issues of BEEBUG itself, I have now finished cataloguing the software from volume 2 but am still having trouble getting hold of all of volume 1 on disc. I have collected a disc containing selected programs from volume 1, including all of issue 10, but if anyone can help with the rest, I would be most grateful.

MAD RABBIT

I have also recently been in contact with Joel Rowbottom, of Mad Rabbit PD (mentioned recently in BEEBUG), and we have swapped software to make each of our libraries more complete. The software that he has kindly sent to me, all of which I had not seen before, includes a collection of *AMPLE* software for the *Music 5000* add-on, some original Basic music programs (so there are no copyright problems) and discs of text files of interest to Star Trek and Red Dwarf fans. There is some entertaining material on these last discs, although I found some of the humour on the Star Trek disc a bit esoteric, and probably most suited to the real addict.

I also obtained a copy of C. J. Richardson's STD Code Directory. This disc-based set of data has been available for some time, but he has added a controlling program to enable searching by town name or by STD code. I have occasionally used programs such as this to check the part of the country from which items have been offered for sale in classified advertisements (there is a similar, but unrelated, MS DOS based program which is available for the Master 512 co-processor). I am pleased that there is now a version available for the BBC Micro

EDUCATION

Quite a long time ago now, I gained permission to distribute software which was produced as part of the Microelectronics in Education Project. The MEP was an educational project in the early 1980's which resulted in the production of a range of software, primarily for schools, aimed mainly at junior age children. I am therefore now pleased to have four volumes of MEP software. The programs are well produced and are useful for a variety of educational uses with young children.

The do have a definite '80s' feel about them, though

The four volumes comprise thirty different programs in all, and cover a wide range of subjects, although they follow no particular sequence or theme. All of the programs work with disc systems, but were written at a time when disc drives were an expensive rarity, in 1983. Programs which use data files, therefore, expect to be using tape. Some which I have tested seem to work with DFS and ADFS data files, but others do select *TAPE, although they should not be too difficult to modify (by removing the *TAPE commands). I will give a few examples, because educational software seems to be a particular interest for a number of readers

Volume 1 contains *Crash*, which requires the use of counting and logic to solve a problem; *Shopping*, where the child must buy five items, from a choice on a shopping list, within a budget; *Quiz*, a multiple choice quiz package, complete with a quiz designer; *Animal*, a simple artificial intelligence program that learns from the answers given to questions and builds up a knowledge base, and Farmer, a version of the old logic problem which involves a farmer who has to get a hen, a dog and a sack of corn across a river; and others.

Volume 2 contains a similarly varied collection. Amongst others, there are: *Eureka*, a simulation of a bath, where the child has to perform a series of actions in a logically thought out order; *Fraction Snap*, a game for practising fractions; and *VennMan* and *VennKid*, two programs designed to help with the learning of sets

In volume 3, I particularly noticed the program, *Brickup*, which is designed to help children to become familiar with the use of a dictionary. Volume 4 includes *Spike*, which helps with early number work by asking the child to count in units, tens and hundreds, *Box*, which helps with the recognition of word shapes; and *Symmetry*, which provides practice in the recognition of lines of symmetry in a 2 dimensional shape.

These are just examples from the four volumes. All in all, I thought the set was well worth a look if you have children in the age range 7 - 11

I have seen quite a lot of similar educational software, seemingly designed for school use, but possibly of interest to parents. Unfortunately, I haven't been able to track down anyone who can tell me if it would be legitimate to circulate copies. Much of the software seems to have been written by individual teachers to help with the teaching of particular points, but there are also many other packages, which give no sign of copyright or of commercial links, that may be of interest to a wider audience. If anyone can help with the identification of such software and with the permission to distribute it, I would be pleased to hear from them.

ADDRESS

I have heard from other contributors to BEEBUG that some readers have contacted them asking for my address. I thought that it was fairly well known by now, but if you can help with the search for volume 1 of BEEBUG or for educational software, or if you want to get in touch with BBC PD or myself, in relation to this column, my address is:

Alan Blundell,
18 Carlton Close, Blackrod,
BOLTON, BL6 5DL.

Next issue, I plan to cover more of the software which I have recently received

A Room with a View

Kieren Hollingsworth gives some hints to View users.

For me, *View* has stood the test of time. Despite taking possession of a PC and *Wordperfect*, I still turn to my trusty Beeb when it comes to typing. However, over the years I have used *View*, I have encountered several little annoyances and devised my own ways of getting round these, which I shall outline here.

EIGHTY CHARACTER MODES

One of the attractions of *View* is that it allows you to type in and view text in eighty character modes (really 74 characters allowing for command spaces), as long as enough memory is available. Therefore, it is often convenient to create your text in mode 3 and use the standard ruler for this mode. This is achieved best by typing MODE 3 (MODE 131 on a Master) followed by *WORD. The word processor will then assume the desired ruler. Entering the commands the other way round will result in the ruler for the previously used mode appearing. This can be corrected by typing NEW before beginning to type.

On a standard television set, the small characters can be rather blurred and difficult to read. Switching the colour off on the set will sharpen them. Another way of improving readability is to change the colour of the display: the easiest way to do this is to issue a VDU19 command to substitute different colours for the standard black and white. This can be done by moving to the command screen and using a series of Ctrl-key combinations with letters on the keyboard. For example, to create black writing on a white background, the ordinary commands would be (in mode 3)

```
VDU19,0,7,0,0,0
VDU19,1,0,0,0,0
```

For these do the following:

```
Ctrl-S-0-7-0-0-0
Ctrl-S-1-0-0-0-0
```

The numbers are the standard colour numbers used by the Beeb. The colour will not change until the last Ctrl-0 has been pressed. Thus to create blue text on a yellow background, use the sequence:

```
Ctrl-S-0-3-0-0-0
Ctrl-S-1-4-0-0-0
```

SET STANDARDS

Once you've found a format that suits you, and you've created your own ruler, it is a good idea to save it by itself, and recall it at the beginning of each new piece of text. Once loaded, use the NAME command to change the default name to the name you will want to save the complete text under, such that simple SAVE commands can be regularly issued to record alterations and guard against outside gremlins. A note of warning: even if the ruler is reloaded, View always switches formatting and justification back on regardless of their state at the time of saving: this can be annoying if you find your text justified when it is not required

GOING BOLDLY ONWARDS

The *bold* and *underline* features are quite simple to use, but a few points need to

be borne in mind, firstly, that underlining will cancel itself automatically at the end of each line whereas the bold will remain active until a second bold symbol is encountered. Forgetting this will waste paper! Also, although the symbols take up a character space on the screen, the computer makes an adjustment at the end of each line to achieve the correct line length. The default printer driver also underlines spaces.

Therefore, when creating tables, write the table on screen with headings, then use the insert mode (Ctrl-f4) to add spaces in front of or after those headings for which the codes are required, and then insert the codes. The spaces just entered will disappear. It gives a stuttered effect on screen but the printout will be correct.

CHECK YOUR SPELLING CHECKER

The following problems are ones I have encountered using Computer Concept's *Spellmaster* with *View*, although the following comments may well apply to other checkers. Sometimes, when memory is tight, the computer will "hang" during spell checking and there is no alternative but to destroy the text, so save it first (and often)!

Another thing to be aware of is that after the spelling checker has substituted words of a different length to the original misspelling, the line may no longer be formatted or justified correctly. The most comprehensive way to deal with this is to complete the spelling check and then start at the beginning of the text and continually press FORMAT PARAGRAPH (f0) through to the end. Tables should be skipped over manually or the alignment of the table could be ruined

DRIVING YOUR PRINTER

View's extended features such as *bold* and *underline* can only be used with a printer driver tailored for the individual printer. The standard driver provided with *View* Version 3.0, *FX4*, allows use of bold and underline on most printers. For other facilities, it is likely that a driver will have to be created using the supplied program. For Citizen 120-D users, the driver published in BEEBUG Vol.5 No.9 seems to work best.

Readers may also like to refer to the View Printer Driver published in last month's BEEBUG (Vol.12 No.4). This driver emulates the Epson LQ-850, and can be readily customised to work with most printers attached to the Beeb.

Another thing that may annoy is *View's* insistence on leaving large spaces at the top and the bottom of every page. To get rid of these, enter editing mode, create four blank lines at the top of the document (using f6) and enter the following commands using Shift-f8

```
TM 0
HM 0
FM 0
BM 0
```

This will then print the whole page. You could of course add these commands to your template file, as discussed earlier in the article, so that they are always used when you create a new document.

Happy Viewing!

the installed software. Despite the result though, archiving of applications is still worthwhile as we'll see when we look at recovery.

DEVELOPING A SYSTEM

Securing a single directory is so simple it doesn't justify a batch file, but when things get more complicated this is worthwhile. Next month I'll explain how ES back-ups work, using parameter-driven batch files. If you didn't realise it, this means I'm leaving you with a little puzzle.

No problem? OK, try this. Create a batch file *MAIN.BAT* containing the following lines.

```
%1
%2
%3
```

Now create three more, *ONE.BAT*, *TWO.BAT* and *THREE.BAT*, each of which simply announces itself on screen. ONE.BAT should contain *ECHO This is ONE.BAT* and so on. Now enter:

```
MAIN ONE TWO THREE
```

You might expect that, with these parameters, *MAIN* will call *ONE.BAT*, followed by *TWO.BAT* then *THREE.BAT*. It doesn't! The puzzle is to see if you can make *MAIN* work before next month's Forum. Oh yes, I almost forgot! *MAIN* must be able to run any combination of *ONE.BAT*, *TWO.BAT* and *THREE.BAT*, from one to all three, in any order.

Add or change anything (except the names) but use only standard batch commands. Extra files (batch or otherwise) programs and manual interference are not permitted. *See you next month!*

Census (Part 3)

This month Paul Goldsmith displays your frequency distributions.

Before these programs can be used it is necessary to go back to the text file *<code>qtx* and amend it by changing QUE to HED and shortening the answers to 7 characters only. This requires a little ingenuity but you need to change it to something like this.

```
SHOP
HED
Shopping Survey
▮
17
200
10
What is the purpose of your visit ?
ShoppnG
Cinema
Theatre
Market
Castle
Museum
Tourism
meal
Bank/BS
HseAgnt
17
```

When the text has been changed, save it as *<code>htx* in straight ASCII as before. Run *CREATE* and select HED instead of QUE and the program will create the required file *<code>HED*. Now type in the program *Freqdi* as listed below. Add *PROCblue* and *PROCt* from the first article and save the program as *Freqdi*

To use *Freqdi* from the *Analyse* menu (option 2), you must first calculate the frequency distributions (option 1). In

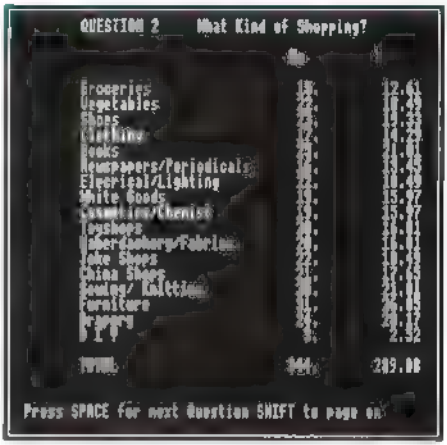


Figure 1. Data presented in text form

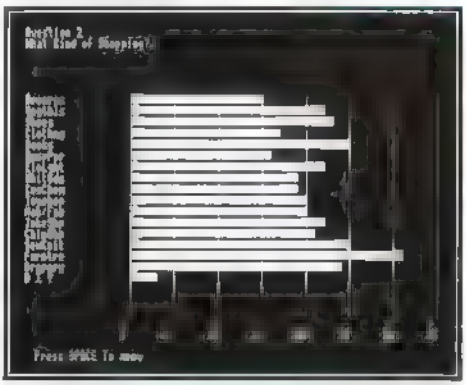


Figure 2. Data as a graph

Freqdi you can have the distributions of the answers to each question printed out, either as text or a graph. You also have the option to save the graph or text to disc for use elsewhere. Apart from the obvious use in a word processor the saved frequencies can be used by *Viewplot* as follows

First *READ* the file into *View* and ensure that there are two blank lines above and below the lines of data. Use Shift-f7 to set markers 1 & 2 at the beginning of the lines, two lines above and two lines below the data. Use *WRITE <filename> 1 2* to save this block using a different file name.

Before saving the block, ensure that there are no spaces in the answers in the left column. These are best filled in with the underline character below the pound sign on the keyboard

Save a separate file for each question and, after running *Viewplot*, select option 1 (Edit Data) and use f5, *Read Spooled File*, to load the file. In reply to "Data format:" press key 5. The Title, Labels Name and Y-axis name must then be added to the screen. Use f3 and enter a filename to save the *Viewplot* Data set

The normal *Viewplot* processes can then be followed. If the plots are in mode 0 it is possible to dump the screen using Ctrl+f8 and read this into a Desktop Publishing program, such as *Wapping Editor*, to add detail or present them on a page. Beebug's *Astaad* can also be used to embellish the screen beforehand.

If a full sideways A4 page print is required, the *Dumpmaster* ROM can be used. This ROM will also drive a colour printer when the plot is in a colour mode.

In next month's final part we will add the cross tabulation display.

```
   10 REM Program Freqdi
   20 REM Version B 1.0
   30 REM Author  Paul Goldsmith
   40 REM BEEBUG  October 1993
   50 REM Program Subject to Copyright
   60 :
  100 REM ONERROR PROCerr
  110 PROCcode
  120 DIMT%(J%),A%(J%),Z%(H%,H%),A$(J%,H
%):Rep$="":VDU15:MODE135:REM MODE 7 for
BBC B
  130 PROCcode
  140 PROCgetparam
  150 PROCdata
  160 MODE 135:PROCt(3,"Frequency Distri
bution"):REM MODE 7 for BBC B
  170 PRINTTAB(8,7)"M E N U "
  180 PRINTTAB(3,10)"DISPLAY FREQUENCY D
ISTRIBUTION  1"
  190 PRINTTAB(3,12)"DISPLAY BAR CHART
            2"
  200 PRINTTAB(3,14)"QUIT
            3"
  210 PRINTTAB(3,20);Rep$
  220 PRINTTAB(3,23)"PRESS NUMBER"
  230 *FX 15,1
  240 G=GET
  250 IF G=49 MODE 128:PROCdisp:GOTO 160
.REM MODE 0 for BBC B
  260 IF G=50 PROChist:MODE 128:PROCbar:
GOTO 160:REM MODE 0 for BBC B
  270 IF G=51 CHAIN"ANALYSE"
  280 END
  290 :
 1000 DEF PROCdata
 1010 ONERROROFF:S%=OPENINS$:PROCefile(S
%,S$)
 1020 FORX%=1TOJ%:FORY%=1TOH%:INPUT#S%,Z
%(X%,Y%):NEXT.NEXT
 1030 CLOSE#S%
 1040 ENDPROC
 1050 :
 1060 DEF PROCdisp
 1070 PROCquest
 1080 CLS:PRINTTAB(0,28)"Press SPACE for
 next Question SHIFT to page on":VDU28,0
,27,79,1
 1090 CLS:PRINTTAB(5,5)"PRESS 'P' TO PRI
NTER"TAB(5,7)"PRESS 'S' TO SCREEN"
 1100 *FX15,1
 1110 A=GET
 1120 @%=&0090A
 1130 PRINTSPC(5);Head$
 1140 INPUTTAB(5)"SPOOL TO VIEW ";S$
 1150 IF INSTR("Yy",S$) INPUTTAB(5)"Name
 of spool file ";Sp$
 1160 INPUT"Start at Question ?"st%:INPU
T"End at Question   ?"en%:IF A=80VDU 15
 1170 IF INSTR("Yy",S$) :OSCLI"SPOOL "+S
p$
 1180 IFA=80 VDU2 ELSE VDU 15
 1190 FORX%=st%TOen%:S%=0:PRINT',PRINTSP
C(7);"QUESTION ";X%,SPC(5);A$(X%,0):PRIN
T:FORY%=1TOA%(X%):S%=S%+Z%(X%,Y%):NEXT
```

```
 1200 PRINTSPC(34);"No";SPC(9);"%":PRINT
 1210 FORY%=1TOA%(X%):P=100*Z%(X%,Y%)/Nu
mb%:PRINTSPC(7);A$(X%,Y%);SPC(23-LEN(A$(
X%,Y%)));:@%=&20008:PRINTZ%(X%,Y%);:@% &
20208.PRINTSPC(5)P
 1220 NEXT:PRINT
 1230 PRINTSPC(7);"TOTAL";:@% &20008:PRI
NTSPC(18)S%, @%=&20208:PRINTSPC(5)(S%/Nu
mb%*10000)/100
 1240 REPEATUNTILGET=32:IF A<>80 CLS
 1250 @%=&0090A
 1260 NEXT:PRINT":PROCtime:VDU3
 1270  IF INSTR("Yy",S$) S$="" Sp$="":*S
POOL
 1280 PRINT''"Press SPACE TO Menu"
 1290 REPEATUNTILGET=32:VDU26
 1300 ENDPROC
 1310 :
 1320 DEF PROChist
 1330 PROCned
 1340 CLS:PROCt(10,"B A R  C H A R T")
 1350 PRINTTAB(5,22)"TYPE NUMBER THEN 'R
ETURN ":INPUTTAB(5,10)"Which question "Q
1%
 1360 INPUTTAB(5,12)"SAVE SCREEN ";A$
 1370 IF INSTR("yY",A$) INPUTTAB(5,14)"S
creen name ";N$
 1380 ENDPROC
 1390 :
 1400 DEF PROCbar
 1410 CLS:PRINT''"Question ";Q1%'A$(Q1%,0
):PRINT
 1420 FORX%=1TOA%(Q1%):PRINTTAB(0,X%+6);
A$(Q1%,X%):NEXT:PROCplot:IF G=50 PROCspa
ce
 1430 *FX15,1
 1440 REPEATUNTILGET=32:VDU20
 1450 IF INSTR("Yy",A$) OSCLI"SAVE "+N$+
" 3000 8000"
 1460 ENDPROC
 1470 :
 1480 DEF PROCplot
 1490 PROCunit:hpos%=300
 1500 vpos%=860
 1510 factor%=125/unit
 1520 PROCscale
 1530 FORN%=1 TO A%(Q1%)
 1540 MOVE hpos%,vpos%-N%*32-32:PLOT1,0,
-20:PLOT81,Z%(Q1%,N%)*factor%,20:PLOT81,
0,-20
 1550 NEXT
 1560 ENDPROC
 1570 :
 1580 DEF PROCunit
 1590 M%=0
 1600 FORN%=1TOA%(Q1%)
 1610 IF Z%(Q1%,N%)>M% M%=Z%(Q1%,N%)
 1620 NEXT:unit=M%/8
 1630 IF unit<=1 unit=1
 1640 IF unit>1 AND unit<2 unit=2
 1650 IF unit>2 AND unit<5 unit=5
 1660 IF unit>5 AND unit<10 unit=10
 1670 IF unit>10 AND unit<20 unit=20
 1680 IF unit>20 AND unit<50 unit=50
 1690 IF unit>50 unit=100
 1700 ENDPROC
 1710 :
 1720 DEF PROCscale
 1730 VDU5:FORX%=0TO7
 1740 MOVE hpos%+X%*125-30,120:PRINT;X%*
unit:MOVE hpos%+X%*125,160:DRAW hpos%+X%
*125,vpos%-64:NEXT:VDU4
 1750 ENDPROC
 1760 :
 1770 DEF PROCspace
 1780 PRINTTAB(2,30);"Press SPACE To men
u"
 1790 ENDPROC
 1800 :
 1810 DEF PROCtime
 1820 *TIME  REM MASTER only
 1830 ENDPROC
 1840 :
 1850 DEF PROCquest
 1860 ONERROROFF:B=OPENIN G$:PROCefile,B
,G$)
 1870 INPUT#B,Head$,J%:FOR X%=1 TO J%:IN
PUT#B,A%(X%):FORY%=0 TO A%(X%):INPUT#B,A
$(X%,Y%):NEXT:NEXT
 1880 CLOSE#B:ENDPROC
 1890 :
 1900 DEF PROCgetparam
 1910 ONERROROFF:C%=OPENINP$:PROCefile(C
%,P$)
 1920 INPUT#C%,Start%:INPUT#C%,End%:INPU
T#C%,Numb%:INPUT#C%,Top%:INPUT#C%,Rmax%
 1930 CLOSE#C%
```

Continued on page 44

BEEBUG WORKSHOP

Text Compression

by David Peckett

This month the Workshop looks at one way round the Beeb Adventurer's problem of insufficient memory. Indeed, in any circumstances where you want to store a lot of text, the 25K or so that is normally available soon gets used up. There is, however, a way of packing text so that you can store up to 50% more in any given space.

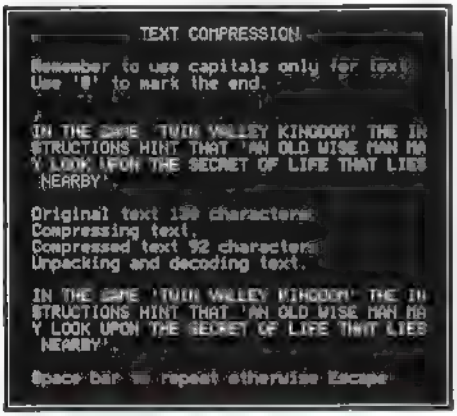
First, though, a quick resume of how characters are stored in the Beeb and most other small computers. Each character is allocated one of 256 so-called ASCII codes in the range 0-255, and each code fits neatly into an 8-bit byte. There is also a variable overhead, of at least 5 bytes, for each string of characters in the Beeb.

This is rather wasteful. If you're only displaying text, you don't need 256 characters and can get by with around 60: A-Z, 0-9, plus some punctuation marks. The exact mixture depends on just what text you have Now, 5 bits can store 32 different character codes and there are 3 groups of 5 bits in two bytes.

However, if we limit ourselves to only 30 characters we can pack three into every two bytes. Unfortunately, 30 is a bit too limiting, so use 60, split them into two groups, and allocate a special code to switch between the groups. In fact, in the early days of computing, this is just how characters were stored as normal on some mainframe computers. Once this is done, all that's left on the Beeb is to find a way of packing three 5-bit codes into two bytes. That is exactly what this first program will do.

```
1000 DEF PROCpackinit
1010 DIM Key$(1),cc%(3)
1020 DIM txtbuf 10000
1030 Key$(0)=
     " ABCDEFGHIJKLMNOPQRSTUVWXYZ.,"
     +CHR$(13)
1040 Key$(1)=
     " 0123456789*;/-=<>()'?$`_;:,"
     +CHR$(13)
1050 ENDPROC
2000 DEF FNcode(str$,locn%)
2010 LOCAL bitptr%,chcode%,chptr%,
     code%,kptr%,lenstr%,ptr%
2020 chptr%=1
2030 lenstr%=LEN(str$)
2040 REPEAT
2050   chcode%=FNcd(MID$(str$,chptr%,1))
2060   code%=code% OR
       chcode%*2^(10-bitptr%*5)
2070   bitptr%=(bitptr%+1) MOD 3
2080   chptr%=chptr%+1
2090   IF bitptr%=0 THEN PROCpack
2100   UNTIL chptr%=lenstr%+2
2110 IF bitptr% THEN PROCpack
2120 =locn%+ptr%
3000 DEF FNcd(ch$)
3010 LOCAL cc%
3020 IF chptr%=lenstr%+1 THEN =0
3030 cc%=INSTR(Key$(kptr%),ch$)
```

```
3040 IF [illegible] THEN [illegible]
3050 ?kptr% = kptr%+1 : ?[illegible]
3060 IF INSTR(key$(k%),ch$) THEN
     ?hptr% = hptr% - [illegible]% - 1 ELSE
     ?kptr% = kptr%+1 : ?[illegible]
3070 [illegible]
4000 DEF PROCpack
4010 locn%?ptr% = code% AND &FF
4020 locn%?(ptr%+1) = code% DIV &100
4030 ptr% = ptr%+2
4040 [illegible]
4050 ENDPROC
```



Example of text compression and decompression

PROCpackinit sets up the system, reserving a buffer area to hold the compressed data - in this example 10000 bytes, but that's up to you. It also sets up coding strings in *Key$(0)* and *Key$(1)*. The two strings here are good general-purpose ones, but you can easily change them to suit your own needs.

Very common punctuation, e.g. space, comma and Return, appears in each. This can help the compression, as there is less need to switch between coding sets. The common characters are in the same place in each string. Also, the two strings each hold a maximum of 30 characters. Although we can code 32 different values into 5 bits, zero is reserved to mark the end of a string while 31 is used as the shift code.

FNcode() packs *str$* into the buffer starting at location *locn%*. Line 2060 packs the characters, in threes, into code% with *FNcd()* extracting the code of each in turn. If the character is not in the currently selected coding string, *FNcd* checks the other; if it finds it, it sends back the shift code (31), otherwise it sets a space. When it reaches the end of the string, *FNcd* returns a zero.

As every three characters are coded, *PROCpack* puts their coded values into the buffer. When it reaches the end of *str$*, any outstanding codes are also stored. On exit, *FNcode* returns the next free address in the buffer. This suits the best way of setting up a set of compressed strings, as in the following pseudo-Basic listing:

```
PROCpackinit
P%=address of start of buffer
REPEAT
  READ a string to code
  P%=FNpack(string,P%)
  UNTIL last string coded
*SAVE buffer to disc
END
```

Don't try to run that! You could read the strings to be packed from DATA statements, the keyboard, or from a disc file; the last is probably the most likely. At the end of the process, your backing store will hold a copy of the compressed text all ready to be unpacked by whatever program is going to use it. If you're writing lots of programs, you'll probably have a standard compression program, to use every time you need it.

The unpacking routines will normally be part of a completely different program from the one which compressed the data.

Before you run the procedure you must first have run *PROCpackinit* and loaded the compressed data into the buffer. Then, by calling *PROCdecode()* with the start address of the packed string, the data will be expanded and displayed on the screen. I leave you to choose the best way of identifying the start of the particular code you need.

```
5000 DEF PROCdecode(locn%)
5010 LOCAL cn%,code%,i%,kptr%,ptr%
5020 REPEAT
5030   PROCunpack
5040   FOR i%=1 TO 3
5050     cn%=cc%(i%)
5060     IF cn%>0 AND cn%<31 THEN
         PRINT MID$(Key$(kptr%),cn%,1);
5070     IF cn%=30 THEN VDU10
5080     IF cn%=31 THEN
         kptr%=(kptr%+1) MOD 2
5090     NEXT
5100   UNTIL cc%(1)=0 OR cc%(2=0) OR
       cc%,3)=0
5110 ENDPROC
6000 DEF PROCunpack
6010 code%=locn%!ptr% AND &FFFF
6020 cc%(1)=code% DIV &400
6030 cc%(2)=(code% AND &3E0) DIV &20
6040 cc%,3)=code% AND &1F
6050 ptr%=ptr%+2
6060 ENDPROC
```

PROCdecode makes repeated calls to *PROCunpack*, which reads the next two bytes from the buffer and splits their contents into the corresponding 5-bit codes in *cc%()*. The codes are then used to extract data from *Key$()*, shifting between the two character sets every time the code 31 is encountered. As soon as the procedure reaches a zero code, it exits as that is the end-of-string marker used

The decoding program must have exactly the same strings in *Key$()* as the original packing program. If the strings are different, the most amazing gibberish may appear. There is little point in using these routines unless you need to store a lot of text. They can never quite reach the 50% improvement mark, but get very near if you use a long string which needs as few shift codes as possible.

A complete demonstration of these routines is contained on this month's magazine disc.

This Workshop was first published in BEEBUG Vol.4 No.5. The information has been checked and updated as necessary for compatibility with later machines. B

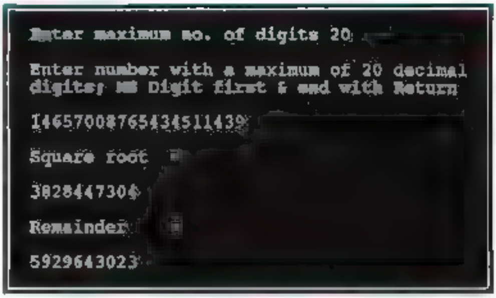
Census

```
1940 ENDPROC
1950 :
1960 DEF PROCcode
1970 ONERROROFF:C%=OPENIN"CODE":PROCefi
le(C%,"CODE")
1980 INPUT#C%,F$,J%,H%,K%:CLOSE#C%
1990 D$-F$+"DA":H$-F$+"HED".G$ F$+"QUE"
Code$-F$+"*****":P$.F$+"PAR"·S$-F$+"DI"
2000 ENDPROC
2010 :
2020 DEF PROChed
2030 ONERROROFF:B-OPENIN H$:PROCefile(B
,H$)
2040 INPUT#B,Head$,J%:FOR X%-1 TO J%:IN
PUT#B,A%(X%):FORY%=0 TO A%(X%):INPUT#B,A
$(X%,Y%):NEXT.NEXT
2050 CLOSE#B·ENDPROC
2060 :
2070 DEF PROCerr
2080 REPORT:PRINT" at line "ERL
2090 PRINT"Press a key":A=GET
2100 ENDPROC
2110 :
2120 DEF PROCefile(I%,I$)
2130 IF I%-0 PRINTTAB(3,19)"File "+I$+"
not found":CLOSE#I%:TIME-0:REPEATUNTIL
TIME=200:CHAIN"ANALYSE"
2140 ENDPROC
```
B

Very Big Numbers (Part 3)

This month Harry Fensom works on square roots

This program finds the integer square root of decimal numbers with lengths up to several thousand digits. The principles used are the same as those in the addition and subtraction programs published in the June edition of BEEBUG (Vol 12 No.2). As before, signs have to be sorted by the user, as has the decimal point, since this is unsigned integer arithmetic



The square root and remainder of a 20 digit number

NOTES ON PROGRAMMING METHODS

Storage and indexing are the same as in the + and - programs. The algorithm used for calculating the square root uses the usual pencil and paper method based on the formula $(a + b)^2 = a^2 + 2*a*b + b^2$ This is not recursive, and has the advantage of giving the nearest exact integer square root of the square number just less than the given number. It is also accompanied by the remainder.

The program performs successive subtractions and shifts, as in the division program (BEEBUG Vol.12 No.3). However, in this case, the subtrahend is twice the current partial root augmented by an extra digit. This is incremented as in the division algorithm until the result goes negative, whereupon we restore the previous result. Because we are not simply using + and -, we cannot use the in-built decimal mode, and so a subroutine is used to do the calculations in decimal and we have to look after the carry implicitly.

USING THE PROGRAM

Enter the listing as given below and save it. When you run it you will be presented with.

```
Enter maximum no. of digits ?
```

The value entered here, say x must take into account the fact that this operation takes longer than those for addition, subtraction and division that you may have used previously, so start with smaller numbers. Follow this number by pressing Return and the display will show:

```
Enter number with a maximum of x
decimal digits, MS Digit first & end
with Return
```

You are entering the number from left to right in decimal. After this, there will be a delay if it is very long, and then the result will be displayed. For a maximum number of digits of 3000, the delay may be of the order of 30 minutes (plus, of course, the time it takes you to type in 3000 digits). The remainder will also be given. Adding the latter to the square of the root obtained gives the original number exactly. This can be useful when we are trying to avoid approximations or rounding errors when solving problems in integers.

As before, for floating point numbers the decimal point has to be fixed by the user For example, to find the square root of 2

to say 1000 decimal places, you might set the max. to say 2010. Then input 2 followed by 2000 zeros. Note that you must use an even number of zeros otherwise you will get the square root of 20 or 0.2 etc. Also note that the root will contain one half, or possibly $\frac{1}{2}x+1$, of the number of digits in the number entered

```
   10 REM Program SQRlarg
   20 REM Version B 4.2c
   30 REM Author  Harry W Fensom
   40 REM BEEBUG  October 1993
   50 REM Program Subject to Copyright
   60 :
  100 INPUT"Enter maximum no. of digits
"m%:max%=m%+2:bytes%=max%/2
  110 PROCinit
  120 PROCassemble
  130 CLS
  140 ?ar%=?opd2p%:ar%?1=opd2p%?1:CALLcl
ear
  150 ?ar%=?resp%:ar%?1=resp%?1:CALLclea
r
  160 ?ar%=?remp%:ar%?1=remp%?1:CALLclea
r
  170 ?ar%=?hd1p%:ar%?1=hd1p%?1:CALLclea
r
  180 ?ar%=?hd2p% ar%?1=hd2p%?1:CALLclea
r
  190 ?tp1%=opd2%MOD256:tp1%?1=opd2%DIV2
56
  200 PROCenter("number")
  210 CALLsqr
  220 PROCdisplay("Square root ",resp%)
  230 PROCdisplay("Remainder ",remp%)
  240 END
  250 :
 1000 DEF PROCinit
 1010 DIMbu% max%,opd2% bytes%,res% byte
s%,rem% bytes%,hd1% bytes%,hd2% bytes%
 1020 DIMmc% &520
 1030 opd2p%=&72
 1040 bup%=&74
 1050 tp1%=&76
 1060 tp2%=&78
 1070 tp3%=&7A
 1080 len%=&7C
 1090 mz%=&7E
 1100 x%=&80
 1110 flag%=&82
 1120 resp%=&84
 1130 cnt%=&86
 1140 ar%=&88
 1150 rev%=&8A
 1160 aux%=&8C
 1170 y%=&8E
 1180 cy%=&90
 1190 hd1p%=&92
 1200 hd2p%=&94
 1210 mx%=&96
 1220 t%=&98
 1230 remp%=&9A
 1240 f%=&9C
 1250 re%=&9E
 1260 ddq%=&F8
 1270 oswrch=&FFEE
 1280 ?opd2p%=opd2%MOD256:opd2p%?1=opd2%
DIV256
 1290 ?bup%=bu%MOD256:bup%?1=bu%DIV256
 1300 ?resp%=res%MOD256:resp%?1=res%DIV2
56
 1310 ?remp%=rem%MOD256:remp%?1=rem%DIV2
56
 1320 ?mz%=bytes%MOD256:mz%?1=bytes%DIV2
56
 1330 ?mx%=max%MOD256 mx%?1=max%DIV256
 1340 ?hd1p%=hd1%MOD256:hd1p%?1=hd1%DIV2
56
 1350 ?hd2p%=hd2%MOD256:hd2p%?1=hd2%DIV2
56
 1360 ENDPROC
 1370
 2000 DEF PROCenter(N$)
 2010 PRINT''"Enter "N$" with a maximum o
f ";m%;" decimal digits;"'" MS Digit fir
st & end with Return"
 2020 ?ar%=?bup%:ar%?1=bup%?1:CALLclear
 2030 I%=0
 2040 REPEAT
 2050 A1%=GET-48:IFA1%>79I%=I% 1:VDU127:
GOTO2050
 2060 IFA1%<10ANDA1%>-1PRINT;A1%;:I%?bu%
=A1%
```

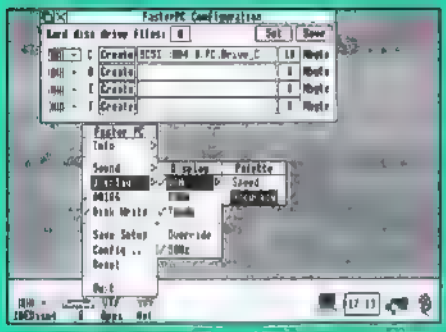
```
2070 I%-I%+1
2080 UNTILI%-max%ORA1%=-35
2090 IFI%-1GOTO130
2100 ?len%-(I% 2)MOD256:len%?1-(I%-2)DI
V256
2110 ?bup%-bu%MOD256:bup%?1-bu%DIV256
2120 CALLinpt
2130 ENDPROC
2140 :
3000 DEF PROCdisplay(N$,T%)
3010 IFN$<>""PRINT'N$" = ""
3020 ?ar%=?T%:ar%?1=T%?1
3030 CALLdisp
3130 PRINT'
3140 ENDPROC
3150 :
4000 DEF PROCassemble
4010 FORpass%=0TO2STEP2
4020 P%=mc%:[OPTpass%
4030 .inpt LDY#0
4040 LDAlen%:STAcnt%
4050 LDAlen%+1:STAcnt%+1
4060 CLC:LDAbup%:ADCcnt%-STAbup%
4070 LDAbup%+1:ADCcnt%+1:STAbup%+1
4080 .iloop LDA(bup%),Y:STA(tp1%),Y
4090 SEC:LDAcnt%:SBC#1:STAcnt%
4100 LDAcnt%+1:SBC#0:STAcnt%+1
4110 BCCiexit
4120 SEC:LDAbup%:SBC#1:STAbup%
4130 LDAbup%+1:SBC#0-STAbup%+1
4140 LDA(bup%),Y
4150 ASLA:ASLA:ASLA:ASLA
4160 ORA(tp1%),Y:STA(tp1%),Y
4170 CLC:LDAtp1%:ADC#1:STAtp1%
4180 LDAtp1%+1:ADC#0-STAtp1%+1
4190 SEC:LDAbup%:SBC#1:STAbup%
4200 LDAbup%+1:SBC#0:STAbup%+1
4210 SEC:LDAcnt%:SBC#1.STAcnt%
4220 LDAcnt%+1:SBC#0-STAcnt%+1
4230 BCSiloop
4240 .iexit RTS
4250 :
4300 .disp LDY#0:LDX#0
4310 LDAmz%:STAcnt%:LDAmz%+1:STAcnt%+1
4320 .brd CLC:LDAar%:ADCcnt%-STAtp1%-ST
Atp2%
4330 LDAar%+1:ADCcnt%+1:STAtp1%+1:STAtp
2%+1
4340 LDA(tp1%),Y:LSRA:LSRA:LSRA:LSRA
4350 CPX#1:BEQpnt
4360 CMP#0:BEQnxt1:LDX#1
4370 .pnt CLC:ADC#48:JSRoswrch
4380 .nxt1 LDA(tp2%),Y:AND#&F
4390 CPX#1:BEQpnt2
4400 CMP#0:BEQrpt:LDX#1
4410 .pnt2 CLC:ADC#48:JSRoswrch
4420 .rpt SEC:LDAcnt%:SBC#1-STAcnt%
4430 LDAcnt%+1:SBC#0.STAcnt%+1
4440 BITcnt%+1:BPLbrd
4450 TXA-BNEdexit
4460 LDA#48:JSRoswrch
4470 .dexit RTS
4480 :
4500 .clear LDY#0
4510 LDAmz%.STAcnt%:LDAmz%+1:STAcnt%+1
4520 .brc CLC:LDAar%:ADCcnt%:STAtp2%
4530 LDAar%+1:ADCcnt%+1:STAtp2%+1
4540 LDA#0.STA(tp2%),Y
4550 SBC:LDAcnt%:SBC#1:STAcnt%
4560 LDAcnt%+1":SBC#0:STAcnt%+1
4570 BITcnt%+1:BPLbrc
4580 RTS
4590 :
5000 .sqr LDY#0
5010 LDAmz%:STAcnt%:LDAmz%+1:STAcnt%+1
5020 LDAmx%:STAx%-LDAmx%+1:STAx%+1
5030 LDA#0:STAy%:STAy%+1
5040 .lb2 CLC LDAopd2p%-ADCy%-STAtp3%
5050 LDAopd2p%+1:ADCy%+1-STAtp3%+1
5060 CLC:LDAbup% ADCx%:STArev%
5070 LDAbup%+1:ADCx%+1:STArev%+1
5080 LDA(tp3%),Y:STA(rev%),Y
5090 CLC:LDAy%:ADC#1:STAy%
5100 LDAy%+1:ADC#0:STAy%+1
5110 SBC:LDAx%:SBC#1:STAx%
5120 LDAx%+1:SBC#0:STAx%+1
5130 LDAmz%:CMPy%:LDAmz%+1:SBCy%+1:BCS1
b2
5140 .lpz CLC:LDAbup%.ADCx%:STArev%
5150 LDAbup%+1:ADCx%+1:STArev%+1
5160 LDA#0:STA(rev%),Y
5170 SEC:LDAx%:SBC#1:STAx%:LDAx%+1-SBC#
0:STAx%+1
5180 BITx%+1:BPLlpz
5190 STYt%:STYt%+1
5200 CLC:LDAmz%-STAcnt%:LDAmz%+1-STAcnt
```

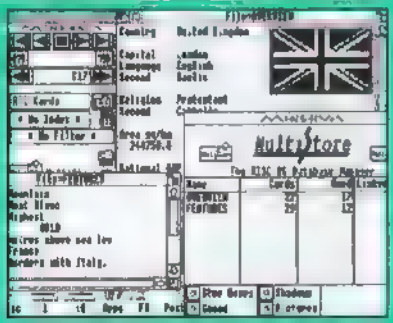
```
%+1
 5210 .clr:CLC:LDAopd2p%:ADCcnt%:STAtp3%
 5220 LDAopd2p%+1:ADCcnt%+1:STAtp3%+1
 5230 LDA#0:STA(tp3%),Y
 5240 SEC.LDAcnt%:SBC#1:STAcnt%
 5250 LDAcnt%+1.SBC#0 STAcnt%+1
 5260 BITcnt%+1:BPLclr
 5270 .jrot JSRrot
 5280 JSRsbth
 5290 CLC:LDAt%:ADC#1:STAt%
 5300 LDAt%+1:ADC#0·STAt%+1
 5310 LDAt%:CMPmz%:LDAt%+1:SBCmz%+1:BCCj
rot
 5320 LDAmz%:STAt%:LDAmz%+1:STAt%+1
 5330 CLC.LDAhd1p%:ADCmz%:STAtp3%
 5340 LDAhd1p%+1:ADCmz%+1:STAtp3%+1
 5350 CLC:LDAbup%:ADCmz%:STArev%
 5360 LDAbup%+1:ADCmz%+1:STArev%+1
 5370 LDAresp%:STAcnt%:LDAresp%+1:STAcnt
%+1
 5380 LDAremp%:STAy%:LDAremp%+1:STAy%+1
 5390 .rvbc:LDA(tp3%),Y:STA(cnt%),Y
 5400 LDA(rev%),Y:STA(y%),Y
 5410 SEC.LDAtp3%:SBC#1:STAtp3%
 5420 LDAtp3%+1:SBC#0:STAtp3%+1
 5430 SEC:LDArev%.SBC#1:STArev%
 5440 LDArev%+1:SBC#0:STArev%+1
 5450 CLC:LDAcnt%:ADC#1:STAcnt%
 5460 LDAcnt%+1:ADC#0:STAcnt%+1
 5470 CLC:LDAy%:ADC#1:STAy%
 5480 LDAy%+1:ADC#0:STAy%+1
 5490 SEC.LDAt%.SBC#1:STAt%
 5500 LDAt%+1:SBC#0:STAt%+1
 5510 BITt%+1:BPLrvbc
 5520 SEC:LDAlen%:SBC#1:STAlen%
 5530 LDAlen%+1:SBC#0:STAlen%+1
 5540 RTS
 5550 :
 5560 .rot
 5570 STYcnt%:STYcnt%+1:STYy%+1
 5580 LDA#1:STAy%:LDA(bup%),Y:STAtp2%
 5590 .rbcl CLC:LDAbup%:ADCy%:STArev%
 5600 LDAbup%+1:ADCy%+1:STArev%+1
 5610 CLC:LDAbup%:ADCcnt%:STAtp3%
 5620 LDAbup%+1:ADCcnt%+1:STAtp3%+1
 5630 LDA(rev%),Y:STA(tp3%),Y
 5640 CLC:LDAcnt%·ADC#1:STAcnt%
 5650 LDAcnt%+1:ADC#0:STAcnt%+1
 5660 CLC:LDAy%:ADC#1:STAy%
 5670 LDAy%+1:ADC#0:STAy%+1
 5680 LDAmx%:CMPy%:LDAmx%+1:SBCy%+1:BCSr
bcl
 5690 LDAtp2%:STA(rev%),Y
 5700 RTS
 5710 :
 5720 .sbth CLC:LDA#1:STAf%:STYa…x%
 5730 .sbc3:STYcy%:LDAf%:STAddq%
 5740 JSRmod
 5750 CLC:LDAopd2p%:ADCmz%:STAtp3%
 5760 LDAopd2p%+1:ADCmz%+1:STAtp3%+1
 5770 LDAre%:STA(tp3%),Y
 5780 CLC:LDAhd1p%:ADCmz%:STArev%
 5790 LDAhd1p%+1:ADCmz%+1:STArev%+1
 5800 LDA(rev%),Y:AND#&0F:ASLA
 5810 CLC:ADCcy%:CLC:ADCddq%:STAddq%
 5820 JSRmod
 5830 LDAre%·ASLA:ASLA:ASLA:ASLA
 5840 ORA(tp3%),Y.STA(tp3%),Y
 5850 LDAddq%:STAcy%
 5860 SEC:LDAmz%:STAtp1%.SBC#1:STAcnt%
 5870 LDAmz%+1:STAtp1%+1:SBC#0:STAcnt%+1
 5880 CLC·LDAhd1p%:ADCtp1%:STArev%
 5890 LDAhd1p%+1:ADCtp1%+1:STArev%+1
 5900 .sbc4 CLC:LDAopd2p%:ADCcnt%:STAtp3
%
 5910 LDAopd2p%+1 ADCcnt%+1:STAtp3%+1
 5920 LDA rev%),Y.AND#&F0
 5930 LSRA:LSRA:LSRA
 5940 CLC:ADCcy%:STAddq%
 5950 JSRmod
 5960 LDAre%:STA(tp3%),Y:LDAddq%:STAcy%
 5970 SEC:LDArev%:SBC#1 STArev%
 5980 LDArev%+1:SBC#0 STArev%+1
 5990 LDA(rev%),Y:AND#&F:ASLA
 6000 CLC:ADCcy%:STAddq%
 6010 JSRmod
 6020 LDAre%:ASLA:ASLA:ASLA:ASLA
 6030 ORA(tp3%),Y:STA(tp3%),Y
 6040 LDAddq%:STAcy%
 6050 SEC·LDAcnt%:SBC#1:STAcnt%
 6060 LDAcnt%+1:SBC#0:STAcnt%+1
 6070 BITcnt%+1:BMInoj
 6080 JMPsbc4
 6090 .noj JSRsubsq
 6100 LDAf%
 6110 CLC:ADC#1:LSRA:STAaux%
```

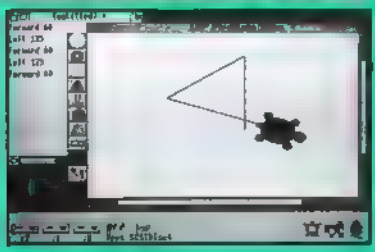
```
 6120 LDAx%,BNEcont
 6130 JSRrstor
 6140 LDXaux%:DEX:STXaux%
 6150 JMPmore
 6160 .cont LDAf%
 6170 CLC:ADC#2:STAf%:CMP#18:BCSmore
 6180 JMPsbc3
 6190 .more LDAmz%:STAcnt%:LDAmz%+1:STAc
nt%+1
 6200 LDAhd1p%:STAtp3%:STArev%
 6210 LDAhd1p%+1:STAtp3%+1:STArev%+1
 6220 .sbc6 CLC:LDArev%:ADC#1:STArev%
 6230 LDArev%+1:ADC#0:STArev%+1
 6240 LDA(rev%),Y:LSRA:LSRA:LSRA:LSRA
 6250 ORA(tp3%),Y:STA(tp3%),Y
 6260 CLC:LDAtp3%,ADC#1,STAtp3%
 6270 LDAtp3%+1:ADC#0:STAtp3%+1
 6280 LDA(rev%),Y:ASLA:ASLA:ASLA:ASLA
 6290 STA(tp3%),Y
 6300 SEC:LDAcnt%,SBC#1:STAcnt%
 6310 LDAcnt%+1,SBC#0:STAcnt%+1
 6320 LDAcnt%:ORAcnt%+1:BNEsbc6
 6330 LDA(rev%),Y:AND#&F0:ORAaux%:STA(re
v%),Y
 6340 RTS
 6350 :
 6360 .subsq LDA#1:STAx%:LDAmz%:STAcnt%
 6370 LDAmz%+1:STAcnt%+1
 6380 .bbc7 CLC:LDAopd2p%:ADCcnt%:STAtp3
%
 6390 LDAopd2p%+1:ADCcnt%+1:STAtp3%+1
 6400 CLC:LDAbup%:ADCcnt%:STArev%
 6410 LDAbup%+1:ADCcnt%+1:STArev%+1
 6420 CLC:LDAnd2p%:ADCcnt%:STAtp2%
 6430 LDAnd2p%+1:ADCcnt%+1:STAtp2%+1
 6440 LDA(tp3%),Y:STAtp1%:LDA(rev%),Y
 6450 SED
 6460 LDXx%:BEQbor:SEC
 6470 JMPgo
 6480 .bor CLC
 6490 .go SBCtp1%:STA(tp2%),Y:BCCbor2
 6500 LDA#1:STAx%
 6510 JMPcont2
 6520 .bor2 STYx%
 6530 .cont2
 6540 CLD
 6550 SEC.LDAcnt%:SBC#1:STAcnt%
 6560 LDAcnt%+1:SBC#0:STAcnt%+1
 6570 LDA#0:BITcnt%+1:BPLbbc7
 6580 CLC:STYcnt%:STYcnt%+1
 6590 .bbc8 CLC:LDAbup%:ADCcnt%:STAtp3%
 6600 LDAbup%+1:ADCcnt%+1:STAtp3%+1
 6610 CLC:LDAhd2p%:ADCcnt%:STArev%
 6620 LDAhd2p%+1:ADCcnt%+1:STArev%+1
 6630 LDA(rev%),Y:STA(tp3%),Y
 6640 CLC:LDAcnt%,ADC#1:STAcnt%
 6650 LDAcnt%+1:ADC#0:STAcnt%+1
 6660 LDAmz%:CMPcnt%,LDAmz%+1 SBCcnt%+1:
BCCbbc8
 6670 RTS
 6680 :
 6690 .mod LDA#0,LDX#8
 6700 .shftsq ASLddq% ROLA CMP#10:BCCnex
t
 6710 SBC#10:INCddq%
 6720 .next DEX:BNEshftsq:STAre%
 6730 RTS
 6740 :
 6750 .rstor STYcy%
 6760 LDAmz%:STAtp2%:LDAmz%+1:STAtp2%+1
 6770 .rbc9 CLC:LDAhd2p%:ADCtp2%:STAtp3%
 6780 LDAhd2p%+1:ADCtp2%+1:STAtp3%+1
 6790 CLC:LDAopd2p% ADCtp2% STArev%
 6800 LDAopd2p%+1:ADCtp2%+1:STArev%+1
 6810 CLC:LDAbup%,ADCtp2%:STAtp1%
 6820 LDAbup%+1:ADCtp2%+1:STAtp1%+1
 6830 SED
 6840 LDXcy%:BNEcar:CLC
 6850 JMPgo2
 6860 .car SEC
 6870 .go2
 6880 LDA(tp3%),Y,ADC(rev%),Y,STA(tp1%),
Y
 6890 BCScar2:STYcy%
 6900 JMPcont3
 6910 .car2 LDA#1:STAcy%
 6920 .cont3
 6930 CLD
 6940 SEC.LDAtp2%:SBC#1,STAtp2%
 6950 LDAtp2%+1:SBC#0:STAtp2%+1
 6960 BITtp2%+1:BPLrbc9
 6970 BPLrbc9
 6980 RTS
 6990 ]:NEXT:ENDPROC
 7000 :
 7010 END
```

RISC user

The number one subscription magazine for the Archimedes







SUBSCRIPTION DETAILS

As a member of BEEBUG you may subscribe to RISC User for the reduced rate of £18.40 - a saving of £1.50 on the normal subscription rate).

Overseas subscription rates are as follows:

£27.50 Europe and Eire, £33.50 Middle East, £36.50 Americas & Africa, £39.50 Elsewhere

RISC User, probably the most popular subscription magazine for the Archimedes, offers all the information you need as an Archimedes user. In every issue of RISC User you will find a wealth of articles and programs with professionally written reviews, lively news, help and advice for beginners and experienced users, and items of home entertainment.

The B5 size of RISC User allows a sophisticated design, big colour illustrations and pages full of information, and yet is still a convenient size to assemble into an easy-to-use reference library. Altogether, in its six years of existence, RISC User has established a reputation for a professional magazine with accurate, objective and informed articles of real practical use to all users of Acorn's range of RISC computers.

Contents of the latest Vol.6 Issue 10 of RISC User:

GROUP SURVEY - DATABASE SYSTEMS

A comprehensive survey of the choices available to users seeking database systems.

NATURE'S WAY: EXPLORING GENETIC ALGORITHMS

An exploration of how ideas of genetics can be used to solve a variety of problems such as that of the classic "Travelling Salesman".

ACORN FLOATING POINT ACCELERATOR

Review of the latest Acorn add-on for the Archimedes giving more power and speed, particularly where scientific calculations are concerned.

FIRSTLOGO FROM LONGMAN LOGOTRON

The latest version of Logo for the educational market from experts Longman Logotron.

FASTER PC: A FASTER PC EMULATOR

A review of an independent software emulation for the Arc of the PC, which claims to be faster than Acorn's offering.

ALL ABOUT FONTS

Acorn's powerful outline font system demystified.

WIMP TOPICS

A major series aimed at readers interested in Wimp programs and Wimp programming. Each article looks at aspects of a particular topic.

WRITE-BACK

The readers' section of RISC User for comment, help, information - a magazine version of a bulletin board.

INTO THE ARC

A regular series for beginners.

TECHNICAL QUERIES

A column which answers your technical queries.

THE DOS SURVIVAL GUIDE

A series of articles on how to use the PC Emulator.

EDUCATIONAL SOFTWARE AT HOME

A series of articles aimed at parents who want to use the Archimedes to help their children in their school work. This issue covers Maths and Numbers.

AUTO-BOOTING THE MASTER 128

D Macgraw

To automatically run a regularly used and never changed Basic (or similar) program, when you switch on your system, blow the program onto EPROM and auto-boot from the RFS (ROM Filing System). Of course, you could also use your battery backed-up sideways RAM, if you're lucky enough to have any

Firstly, write a *!Boot* file in Basic to CHAIN the required program(s). Use Bernard Hill's *BSave* utility (BEEBUG Vol.6 No.7), or David Cox's *Starrun* (BEEBUG Vol 11 No.5), to convert the *!Boot* file into one which can be run with the *RUN command. At the start of the actual program, add a *DISC or *ADFS command to reinstate the (A)DFS. Then use Mark Lock's *ROMGen* (BEEBUG Vol.6 No.6) ROM Generator to put the *!Boot* file and the program into ROM image format.

Once you've blown the image files onto EPROM, or *SRLOADed them into your battery backed-up sideways RAM, you will need to configure your Master to auto-boot from the RFS. Just type the following commands to do this:

```
*CONFIGURE FILE15
*CONFIGURE BOOT
*CONFIGURE LANG12
```

You could also perform various other tasks with the *!Boot* file before loading the main program. For example, it could install the *Enhanced Printer Buffer* (BEEBUG Vol.5 No.7 and Vol.6 No.10) into sideways RAM. However you can't use the *SRLOAD command in a *!Boot* file, as it only loads the first 512 bytes of the file (a MOS 3.20 bug?), so you would need to use the following sequence:

```
*LOAD Buffer E00
*SRWRITE E00+B12 8000 7
?&2A8=&82
*BUFFON
```

to install and activate the buffer before CHAINing the main program.

YET ANOTHER DAY FINDER

Thomas Watts

The following function will print out the day of the week on which any date from January 1st 1780 onward falls. To call it, use a statement in the form:

```
day$=FNday(D,M,Y)
```

where *D*, *M*, and *Y* are the day, month and year respectively.

```
10 DIM M(11),N$(7)
20 DEF FNday(D,M,Y)
30 Y=(Y-1780)*365+INT((Y-1777)/4)-
INT((Y-1701)/100)+INT((Y-1601)/400)
40 IF M=1 M=0:GOTO 100
50 FOR I=1 TO M-1
60 READ M(I)
70 IF M<13 M=M(I) ELSE M=M+M.I)
80 NEXT
90 DATA 31,28,31,30,31,30,31,31
100 DATA 30,31,30
110 N=Y+M+D
120 N=N-INT(N/7)*7
130 RESTORE 160
140 FOR J=1 TO N+1
150 READ N$(J)
160 NEXT
170 DATA FRIDAY,SATURDAY,SUNDAY,MONDAY
180 DATA TUESDAY,WEDNESDAY,THURSDAY
190 =N$(N+1)
```

The function does not perform date validation, and will treat a false entry such as September 31st as October 1st. You could, however, customise it to use David Abbot's date checking routine (BEEBUG Hints and Tips Vol.12 No.1).

Personal Ads

BEEBUG members may advertise unwanted computer hardware and software through personal ads [illegible] possible. Although we will try to include all ads received, we reserve the right to edit or reject any if [illegible]

[illegible]

We [illegible] featured separately [illegible] BEEBUG [illegible]

117 Hatfield Road, St Albans, Herts AL1 4JS

BBC 5.25 games. [illegible] Strykers Run [illegible] Roundheads, Tank Attack £3 [illegible] p&p £2 [illegible] Spycat, Superior Soccer, Airwolf, [illegible] p&p £2. Tel. Colchester [illegible]

Soundcard and Superdump £25 each, Office [illegible] Master £5 each Termulator B+ 12.00, GXR B+ [illegible] and a new BBC [illegible] upgrading, plus 2 cassette games [illegible] write to Mr C Game, 24 Grosvenor, Tiptree, Colchester Essex [illegible]

WANTED PC software for Master 512 [illegible] pay [illegible] First Word Plus, Gem [illegible] write to Mr C Game, 24 Grosvenor, Tiptree, Colchester, Essex CO5 0JN.

WANTED [illegible] disc for Master 128, original and backup lost in accident, local dealers no longer [illegible] Tel. Cornwall [illegible]

WANTED Acorn teletext adapter. Tel. London [illegible]

WANTED Help required with connecting up a BBC [illegible] OS ROM with a Vine Micros Master Replay ROMboard 3, I have the ROM and the switch but don't have any info on what to do with them! Tel. [illegible] 0525 [illegible] ask for Tim after 6pm, [illegible] then ring you back

Make any offer! All items will go to the best offer received within seven days of publication. Master 128 with internal [illegible] co-processor and mouse, complete with Welcome Guide and disc. Microvitec 1451 colour monitor, Cumana double disc drive [illegible] powered, [illegible] Gem discs, Dabs Press Master 512 User Guide [illegible] Panasonic [illegible] printer [illegible] Hybrid Music 5000 synthesiser, Hybrid Music 4000 keyboard with footswitch, Hybrid Music Toolbox disc (all with user-guides and discs), "Play" tutor and disc (cost over £400) £195? BEEBUG Master [illegible] ADFS [illegible] BEEBUG Pro [illegible] disc and guide [illegible] BEEBUG Studio 8 disc and guide [illegible] BEEBUG magazines, complete set to date weights approx [illegible] postage [illegible] BEEBUG programs [illegible] on [illegible] ADFS discs £15. **Wordwise +2 ROM** with segment [illegible] and books [illegible] Access [illegible] discs [illegible] Creative Graphics book, with a DFS graphic disc [illegible] Structured Basic National Extension College [illegible] Freeman £5. **Acorn Master Reference manuals 1&2** £5 each [illegible] West Sussex or transport by arrangement. Tel. [illegible]

Master 128 with [illegible] D/S 3.5 and 40/80 5.25 drives in custom built pine plinth housing, Ferguson colour TV monitor with RGB [illegible] Viewsheet [illegible] spreadsheet, **Viewspell 01 dictionary and Viewstore 01 database in quad cartridge, complete with manuals** and disc [illegible] original [illegible] discs, [illegible] Bank Manager and manual. **Star LC10 9 pin printer. 30hr Basic** [illegible] Guide to View, BEEBUG mag discs [illegible] BEEBUG magazines Vol.7 - Vol.12, also [illegible] BBC Computer User (Watford. Tel. Mid Glamorgan [illegible]

Can I draw members attention to a getting established Beeb User & PD group, named 8 Bit Software c/o Chris Richardson, 17 Lambert Park Road, Hedon, Hull HU12 8HF [illegible] Don't let the BBC die [illegible] recommend you get in touch for a [illegible] 0525 [illegible] after 6pm.

Facit 4511 and Smith Corona D100 printer user handbooks required [illegible] pay costs. Tel. Lancs 061 [illegible] 4591 after 6pm.

Master 128 [illegible] 5.25 drive some software £150, PMS Genie and utils disc [illegible] tape with [illegible] manuals [illegible] disc, £5 Flip' tape £2 [illegible] Daylight disc £5, Exile disc £10 or reasonable offers. Tel. Weston-s-Mare 0934 [illegible]

Electron plus 1 [illegible] disc interface DFS [illegible] View cartridge [illegible] tape [illegible] Cumana disc drive DS 40/80T switch, user guide, Electron User magazines 1987-1989 some [illegible] Electron User tapes £40 the lot. Tel. Oxford [illegible]

Volume No.1 [illegible] Volume [illegible] complete Volume 12 to current issue. Volume [illegible] in BEEBUG binders, also BEEBUG discs from Volume [illegible] to Volume 10.5 (5.25") Volume 10.6 to 12 [illegible] good clean condition sensible offers. Tel. Kent [illegible] evenings only

Micro User [illegible] original [illegible] 40T magazine discs, from July 1989 to July 1990 inclusive £12 or exchange [illegible] software in similar number of discs [illegible] educational software [illegible] software for dyslexics. Tel. Dorset 0305 852276.

Master 128, colour monitor, 2x5.25" drives, many used discs and ROMs [illegible] Tel. Essex [illegible]

Master 128 [illegible] 40T single drive, mouse, Overview, Slogger Press Music System, Wordwise, System Gamma etc. [illegible] games discs and tapes, BEEBUG complete [illegible] books, other software, discs etc. all boxed with manuals £200 the lot [illegible] split. Tel. Derbyshire [illegible] 71535.

Help wanted Ref Crossword Compiler [illegible] Anniversary disc. Can anyone please supply the changes necessary to enable the above program to work correctly on a Star LC24 [illegible] printer [illegible] compatible. The problem is getting the correct vertical [illegible] of the numbering when using the load [illegible] with a previously saved crossword, (followed by the F4 print option and No.2 grid No. option in either size [illegible] seems well. If anyone has a listing that works or can tell me what needs to change to make it work I will be very grateful. Tel. [illegible] 0646 681275.

Master 128 with 65C102 co-processor & user guide parts 1&2, [illegible] disc drive [illegible] power lead [illegible] Acorn cassette recorder all in good condition with [illegible] and [illegible] cables [illegible] Star LC-20 printer, various utility discs, including soft ROMs [illegible] BASIC [illegible] books [illegible] User Guide for BBC Micro, Wordwise + by Bruce Smith and [illegible] years BEEBUG mags [illegible] for £200. Tel. [illegible]

WANTED double sided dual 40/80T disc drive with power supply for BBC Master 128, must be in full working order and reasonably priced. Tel. [illegible] 860842

New unused [illegible] multi-sync monitor [illegible] unwanted part of A5000 system £250 o.n.o. Tel. Uxbridge 0895 635695 eves only

Master 128 RGB colour monitor, 5.25" dual disc drive double plinth, 3 disc storage boxes with 80 blank discs [illegible] educational software for young and older children also lots of original commercial games [illegible] and nearly every Repton game, Typing Tutor and advanced Maths package, complete set of BEEBUG magazines plus other books, software, only £300. Tel. Portsmouth 0705 [illegible]

BBC Master MOS 3.20, Terminal [illegible] View B3 [illegible] ADFS 1.50, Edit 1 [illegible] Viewsheet B [illegible] DFS 2.24, SRAM 1.04, Spellmaster [illegible] Wordwise Plus [illegible] Overview [illegible] 1.53 fitted, cassette, bridge of twin 40/80 5.25 disc drives, Microvitec 653 colour monitor [illegible] and utilities, 512 board fitted with mouse and programs all originals, lots of support material, £375 o.n.o. Tel. Leeds [illegible]

WANTED Games, Educational discs 5.25" for BBC B. Tel. Norfolk [illegible] (work) [illegible] (home)

BBC Master Compact complete with colour monitor and 3.5" [illegible] Overview [illegible] with manuals [illegible] Tel. [illegible] anytime

POSTBAG

WORDWISE AND WORD PROCESSING ON AN ARC

I was surprised to see the assertion in the editorial page of BEEBUG Vol.12 No.3 that "it is perfectly feasible to use old favourites like View, and Wordwise on an Archimedes". I cannot speak for View, which I have always avoided, but I assume that your intention was to refer to Wordwise Plus rather than the original Wordwise. I have reason to think that far from being "perfectly feasible", it is in fact scarcely practical.

A test by a colleague produced the report that Wordwise Plus is not really usable under the BBC emulator because of the slow rate of screen updating. This is disappointing, because I believe that Wordwise Plus cannot be regarded as a serious word processor on the Arc.

My message to Wordwise Plus users, therefore, is stick to an 8-bit machine, preferably a Master 128 or machine with shadow RAM. For typical personal word processing requirements, this combination has proved hard to beat.

Colin Robertson

The reference to "Wordwise" in the editorial was used generically, rather than as a specific identifier. Moreover, it was not our intention to imply that any of the Beeb word processors provided the ideal solution for this task, more that they (and associated files) could continue to be used on an Archimedes. Most users experiences seem to be that despite many misgivings old methods, using files and software from 8-bit machines, are rapidly abandoned in favour of the more powerful and flexible applications written specifically for the Archimedes.

COLOURED RIBBONS

Some time ago BEEBUG carried a piece on multi-coloured printing on a dot-matrix printer using single colour ribbons (Vol.8 No.4). Since then I've been trying to get hold of coloured ribbons (single colours other than black), but with no success, and I have been unable to locate a supplier. Can any readers help?

Chris Robbins

If any readers can provide sources of coloured ribbons, we will publicise this through BEEBUG, and of course let Chris Robbins know.

MAINTAINING A BEEB

I have several times experienced failure with Astec AA11660 power supplies for the Beeb. These supposedly "throw-away" items cost about £60! Electrolytic capacitor C9, rated 220 micro farads at 10 Volts, had decreased to about 10 micro farads in all cases. C9 lies between the power transformer and the TO3 power transistor, and forms part of the "start up" circuit. A "high frequency" electrolytic is required, and a Maplin 220 micro farad 50 Volt capacitor (at about 50p) will just squeeze into the space available. After extracting the PSU (Power Switched Unit) from the computer, you need to free two cable ties, the switch and the power output socket, and extract three mounting screws and two earthing tags, to gain access to the underside of the PCB. I do also recommend that you use a thermostatically regulated soldering iron.

In the event of a faulty PSU, many faults can occur, but replacing C9 is a good cheap first guess.

C.J.Chapman

This is all useful advice, but we would advise caution unless you really do feel competent with a soldering iron. However, if the power unit has failed, and a complete replacement is the only other alternative, then there is little to lose, but do take care or your 'repair' might end up by damaging other parts of your Beeb. Also, do make sure any equipment is disconnected from the mains when being repaired. We can accept no responsibility for the consequences as a result of following Mr.Chapman's advice.

BEEBUG MEMBERSHIP

Send applications for membership renewals, membership queries and orders for back issues to the address below. All membership fees, including overseas, should be in pounds sterling drawn (for cheques) on a UK bank. Members may also subscribe to RISC User at a special reduced rate.

RENEWAL RATES FOR BEEBUG MAGAZINE AND MAGAZINE DISC SUBSCRIPTIONS

See May 1993 Editorial for further explanation

The table below shows the renewal rate applying after the June issue 1993 according to the first issue of the renewal period. For joint BEEBUG/RISC User subscriptions add half the appropriate BEEBUG renewal rate to the full RISC User renewal rate; (UK £18.40, Europe & Eire £27.50, Middle East £33.50, Americas & Africa £36.50, Elsewhere £39.50).

Renewal Issue	Issues to go	Mag UK	Mag Europe	Mag Mid-E	Mag Am+Af	Mag Else	Disc UK	Disc O'Seas
Jun	9	16.56	24.75	30.15	32.85	35.55	45.00	50.40
Jul	8	14.72	22.00	26.80	29.20	31.60	40.00	44.80
A/S	7	12.88	19.25	23.45	25.55	27.65	35.00	39.20
Oct	6	11.04	16.50	20.10	21.90	23.70	30.00	33.60
Nov	5	9.20	13.75	16.75	18.25	19.75	25.00	28.00
Dec	4	7.36	11.00	13.40	14.60	15.80	20.00	22.40
J/F '94	3	5.52	8.25	10.05	10.95	11.85	15.00	16.80
Mar '94	2	3.68	5.50	6.70	7.30	7.90	10.00	11.20
Apr '94	1	1.84	2.75	3.35	3.65	3.95	5.00	5.60

BACK ISSUE PRICES (per issue)

All overseas items are sent airmail. We will accept official UK orders for subscriptions and back issues, but please note that there will be a £1 handling charge for orders under £10 which require an invoice. There is no VAT on magazines.

Volume	Magazine	5"Disc	3.5"Disc
6	£1.00	£3.00	£3.00
7	£1.10	£3.50	£3.50
8	£1.30	£4.00	£4.00
9	£1.60	£4.00	£4.00
10	£1.80	£4.75	£4.75
11	£1.90		£4.75

POST AND PACKING

Magazines and discs are postcode a
Please add the cost of p&p when ordering.
When ordering several items use the highest price code, plus half the price of each subsequent code. UK maximum £8.

Post Code	UK, BFPO Ch.I	Europe, Eire	Americas,Africa, Mid East	Elsewhere
a	£1.00	£1.60	£2.40	£2.80
b	£2.00	£3.00	£5.00	£5.50

BEEBUG
117 Hatfield Road, St.Albans, Herts AL1 4JS
Tel. St.Albans (0727) 840303, FAX: (0727) 860263
Office hours: 9am-5pm Mon-Fri Showroom hours: 9am-5pm Monday to Saturday
(24hr Answerphone for Connect/Access/Visa orders and subscriptions)

BEEBUG MAGAZINE is produced by RISC Developments Ltd.

Editor: Mike Williams
Assistant Editor: Kristina Lucas
Editorial Assistance: Marshal Anderson
Production Assistant: Sheila Stoneman
Advertising: Sarah Shrive
Subscriptions: Helen O'Sullivan
Managing Editor: Sheridan Williams

CONTRIBUTING TO BEEBUG PROGRAMS AND ARTICLES

We are always seeking good quality articles and programs for publication in BEEBUG. All contributions used are paid for at up to £50 per page, but please give us warning of anything substantial that you intend to write. A leaflet 'Notes to Contributors' is available on receipt of an A5 (or larger) SAE.

Please submit your contributions on disc in machine readable form using plain text format if possible for text, but please ensure an adequate written description is also included of your submission and the contents/format of your disc.

In all communication, please quote your membership number.

Printed by Arlon Printers (0923) 268328 ISSN - 0263 - 7561

Magazine Disc

October 1993

CENSUS - This month, in the third instalment of this series on data capture and analysis, we provide the complete suite of census programs with the extensions to allow you to analyse your data.

TEXT COMPRESSION - A demonstration of this month's Workshop routines showing how to fit more text into limited space, using David Peckett's cunning routines.

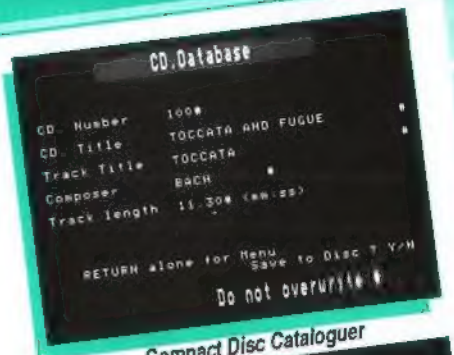
BIG SQUARES - This program from the Big Maths feature, calculates the square roots of very, very large numbers. We have also included two additional demonstrations of the Big Maths routines.

HUNCHBACK RETURNS - Our Golden Oldie this month is a Quasimodo game. Can you dodge arrows and swing over bottomless pits (well! O.K. very deep ones then), to rescue Esmeralda in time. Not forgetting to ring a few bells in the process, of course.

DESIGNER CHARACTERS - Use this multiple character designer to create larger designs and patterns from individual characters. No more fumbling with pens and graph paper. This utility will generate the character set for you, and even gives you the VDU23 commands for use in your own programs.

ORGANISE YOUR COMPACT DISCS - This useful application will bring order to the most unruly of CD boxes. It will find the CD, track, or composer that best suits your mood.

MAGSCAN DATA - Bibliography for this issue of BEEBUG (Vol.12 No.5).



Compact Disc Cataloguer

Multiple Character Designer

Quasimodo

Special BEEBUG Members Offer

SPECIAL OFFERS AND REDUCED PRICES ON NEW ACORN COMPUTERS *PLUS* SPECIAL COMPUTER CONVERSION PACKAGE

If you have been considering converting from your BBC computer to a new Acorn RISC computer this may be the best time. Not only are we at BEEBUG offering fantastic deals on special A3010 packages but Acorn have just announced new computer systems which with their new software bundles offer a considerable price reduction. In addition we are continuing to offer the special conversion package to help you convert systems thus simplifying the transfer of software from your BBC.

We are still offering the **A3010 Colour Family Solution** at **£479** ex VAT £562.82 inc VAT. **Save over £160**. Code 0162g and the **A3010 Home Professional System** for only **£765.10** ex VAT £899 inc. VAT. **Save over £290**. Code 0179g

NEW COMPUTER SYSTEMS FROM ACORN

Acorn have just announced the new line up of systems for Christmas 1993. Although most of the computers themselves have remained unchanged, the prices have come down and there are some exciting new software bundles available with them.

A3010 Action Pack

Code 0203g **£339.57**
(£399 inc VAT)

The entry level system is now available at an incredible £399 inc VAT. This is amazing value. The computer can be plugged straight into your television or used with an optional monitor.

1Mb Memory (Upgradable to 4Mb)
1 Free Mini Expansion Slot
2 Joystick Ports
TV Modulator

The Action pack includes Zool, Startwrite and an Audio Training Tape, along with demonstration versions of Lemmings, Chuck Rock, Superpool and Fervour.

A3010 Learning Curve System

Code 0173g **£637.45**
(£749.00 inc VAT)

Colour Monitor (AKF30)
2 Mb Memory (Upgradable to 4Mb)
1 Free Mini Expansion Slot
2 Joystick Ports
Learning Curve Software Pack

A3020 FD Colour System

Code 0151g **£675.00**
(£793.12 inc VAT)

Colour Monitor (AKF40)
2 Mb Memory expandable to 4Mb
1 Free Mini Expansion Slot

A4000 Colour System

Code 0227g **£850.21**
(£999 inc VAT)

Colour Monitor (AKF40)
2 Mb Memory Upgradable to 4Mb
80 Mb Internal Hard Drive
1 Free Mini Expansion Slot
Option:
Multiscan Monitor (AKF18) **£42.55**

The Home Office Pack

This is available with the A4000 or A5000 and consists of Easiwriter, Datapower, Pipedream 4, PC Emulator, DR DOS 6 and an Audio Training Tape. It also contains demonstration versions of Prophet and Almanac.
Option price: **£85.11**

A5000 Multiscan System

Code 0325 **£1275.74**
(£1499 inc VAT)

Now with the ARM 3 running even faster at 33Mhz

Multiscan Monitor (AKF18)
2 Mb Memory Upgradable to 8Mb
80 Mb Internal Hard Drive
4 Free Full Expansion Slots
Option:
160HD 4Mb RAM System **£170.22**

The Learning Curve Pack

This is available with the A4000 and A5000. It includes Acorn Advance, PC Emulator, DR DOS 6 and an Audio Training Tape. It also includes demonstration versions of Rhapsody II, ScoreDraw, Vox Box, Fervour, Topographer, Darryl the Dragon, smArt, The Crystal Rain Forest, Naughty Stories and ArcVenture.
Option price: **£42.55**

SPECIAL CONVERSION PACKAGE

When you buy a new Acorn computer these various conversion offers are designed to give you excellent value and a reasonably trouble free change to your new faster computer system. The new system will open up new horizons and give you a greater choice of an ever growing number of software packages.

All these offers are backed up by our telesales, technical support and computer repair services. If you require more information about any of the equipment you may need to transfer your system then reference to BEEBUG magazine issues Volume 8 numbers 9 and 10 will help or telephone us directly for technical advice.

SERIAL LINK SOFTWARE TO EASE TRANSFER OF YOUR FILES

So as to cause you as little upheaval as possible when changing your system we are offering you Ivoryash's Serial Link and lead for only £25.

FREE RISC USER SUBSCRIPTION OFFER

(when you purchase one of the above computers)

We will supply you with a subscription to RISC User magazine to run concurrently with your copies of BEEBUG until April 1994, whether or not your subscription ends before that date.

TRADE IN YOUR EXISTING EQUIPMENT FOR A NEW SYSTEM

You may prefer to keep your BBC and sell it privately (don't forget that members may place free ads in BEEBUG magazine). Should you wish us to take it in part exchange we are currently able to offer the following prices:

BBC Issue 7	**£20**	**BBC Issue 7 with DFS**	**£40**
Master 128	**£75**	**Microvitec monitor**	**£20**

Trade in offers can be applied up to two months after purchase of your new equipment. Trade in prices INCLUDE VAT, all others are ex. VAT.

BEEBUG **117 Hatfield Road, St Albans, Herts AL1 4JS**
Tel: 0727 840303 (24 hours) Fax: 0727 860263 All prices are ex VAT and p&p